Machine Transliteration of Proper Names between English and Persian

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Sarvnaz Karimi
BEng. (Hons.), MSc.

School of Computer Science and Information Technology,
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

March, 2008

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Sarvnaz Karimi
School of Computer Science and Information Technology
RMIT University
March, 2008

Acknowledgements

During my PhD candidature, I have had the privilege to work with great supervisors, Dr Andrew Turpin and Dr Falk Scholer, who offered me their trust and patience. Their assistance, advice, and constructive comments and criticism were invaluable to me. I am proud of all the joint work we did during my candidature. To them I owe my deepest gratitude.

My gratitude also goes to Dr Ali Moeini, my masters degree supervisor, who had been a never-ending source of encouragement, which led to my pursuit of a PhD degree.

Many thanks to all of the volunteers who dedicated their time to construct the bilingual corpora I required for my experiments: Behnaz Abdollahi, Naghmeh Alibeik Zade, Shahrzad Bahiraie, Nahid Bozorgkhou, Azam Jalali, Haleh Jenab, Masoumeh Kargar, Mahshid Mehrtash, Mitra Mirzarezaee, Apameh Pour Bakhshandeh, Sahar Saberi, Sima Tarashion, my dear sisters Nastaran and Nazgol, and many others.

The whole journey of my life as a research student at RMIT was made enjoyable by my friends at the search engine group, in particular, Dayang, Pauline, and Ying.

I am also grateful for the international postgraduate research scholarship (IPRS) provided by the Australian government, and also the financial support from the school of Computer Science and Information Technology, RMIT University.

Last but not the least, I would like to dedicate this thesis to my mother for her understanding, encouragement, and immense support; and to the memory of my father for inspiring my fascination in learning and science, and his support in pursuing my studies.

Sarvnaz Karimi

Credits

Portions of the material in this thesis have previously appeared in the following publications:

- Sarvnaz Karimi, Andrew Turpin, Falk Scholer, English to Persian Transliteration, In Proceedings of *13th Symposium on String Processing and Information Retrieval*, Volume 4209 of Lecture Notes in Computer Science, pages 255–266, Glasgow, UK, October 2006.

- Sarvnaz Karimi, Andrew Turpin, Falk Scholer, Corpus Effects on the Evaluation of Automated Transliteration Systems, In Proceedings of *The 45th Annual Meeting of the Association for Computational Linguistics*, pages 640–647, Prague, Czech Republic, June 2007.

- Sarvnaz Karimi, Falk Scholer, Andrew Turpin, Collapsed Consonant and Vowel Models: New Approaches for English-Persian Transliteration and Back-Transliteration, In Proceedings of *The 45th Annual Meeting of the Association for Computational Linguistics*, pages 648–655, Prague, Czech Republic, June 2007.

- Sarvnaz Karimi, Effective Transliteration, *Australasian Computing Doctoral Consortium*, Ballarat, Australia, January 2007.

- Sarvnaz Karimi, Enriching Transliteration Lexicon by Automatic Transliteration Extraction, *HCSNet SummerFest'07: Student Posters*, The University of New South Wales, Sydney, Australia, December 2007.

- Sarvnaz Karimi, Falk Scholer, Andrew Turpin, Combining Transliteration Systems by Black-Box Output Integration, *RMIT University Technical Report* TR-08-01, 2008.

This work was supported by the Australian government Endeavour International Postgraduate Research Scholarships program and the Information Storage, Analysis and Retrieval group at RMIT University.
The thesis was written in the `Vim` editor on Mandriva GNU/Linux, and typeset using the LaTeX 2_ε document preparation system.

Note

Unless otherwise stated, all fractional results have been rounded to the displayed number of decimal figures.

Contents

List of Figures

List of Tables

Abstract

Machine transliteration is the process of automatically transforming a word from a source language to a target language while preserving pronunciation. The transliterated words in the target language are called out-of-dictionary, or sometimes out-of-vocabulary, meaning that they have been borrowed from other languages with a change of script. When a whole text is being translated, for example, then proper nouns and technical terms are subject to transliteration. Machine translation, and other applications which make use of this technology, such as cross-lingual information retrieval and cross-language question answering, deal with the problem of transliteration. Since proper nouns and technical terms — which need phonetical translation — are part of most text documents, transliteration is an important problem to study.

We explore the problem of English to Persian and Persian to English transliteration using methods that work based on the grapheme of the source word. One major problem in handling Persian text is its lack of written short vowels. When transliterating Persian words to English, we need to develop a method of inserting vowels to make them pronounceable. Many different approaches using n-grams are explored and compared in this thesis, and we propose language-specific transliteration methods that improved transliteration accuracy. Our novel approaches use consonant-vowel sequences, and show significant improvements over baseline systems. We also develop a new alignment algorithm, and examine novel techniques to combine systems; approaches which improve the effectiveness of the systems.

We also investigate the properties of bilingual corpora that affect transliteration accuracy. Our experiments suggest that the origin of the source words has a strong effect on the performance of transliteration systems. From the careful analysis of the corpus construction process, we conclude that at least five human transliterators are needed to construct a representative bilingual corpus that is used for the training and testing of transliteration systems.

Chapter 1

Introduction

"If we knew what it was we were doing, it would not be called research, would it?"

— Albert Einstein

1.1 Motivation

Machine translation (MT), the process of automatically translating texts from a *source* natural language to a *target* natural language, has attracted many researchers around the globe for over 50 years. When it was first attempted during World War II, it was considered as a problem that could be solved in two or three years. However, after decades, fully automatic translation is still far from a solved problem [Jurafsky and Martin, 2008], particularly for languages that differ greatly in grammar and script. Apart from the need for automatic translation on its own, many multilingual processing applications also require the automatic translation of texts; such applications include cross-lingual information retrieval and cross-lingual question answering.

Machine transliteration emerged as part of machine translation to deal with proper nouns and technical terms that are translated with preserved pronunciation. Transliteration is a sub-field of computational linguistics, and its language processing requirements make the nature of the task language-specific. Although many studies introduce statistical methods as a general-purpose solution both for translation and transliteration, specific knowledge of the languages under consideration is still beneficial.

Generally, a transliteration system inputs a word in a source language and outputs a

word in a target language which is pronounced the same as the source word. This process is also called phonetical translation. A transliteration system performs the transformation using a transliteration model, created specifically for the source and target languages. Since the existence of such a model is crucial, the transliteration that a system provides becomes specific to the language-pairs on which it is trained.

In this thesis, we focus on fully automatic transliteration of the Persian and English language-pair. Persian is an Indo-European language which uses a modified Arabic script and is spoken in Iran, Tajikistan, Uzbekistan, Afghanistan, by minorities in some of the countries in the south of the Persian Gulf, and some other countries. In total, it is spoken by approximately 134 million people around the world as first or second language[1]. English and Persian have not been previously studied as a language-pair in any machine transliteration study, which was a strong motivation for this thesis. The results of this thesis will assist a machine translation system that might support Persian. It is also helpful in other applications such as for name search in cross-lingual information retrieval, which searches for information related to a particular person, company, product or any other proper name that can be found in both languages, English and Persian, and is queried in either one of them.

1.2 Challenges in Machine Transliteration

The main challenges that machine transliteration systems encounter can be divided into two categories: language script and evaluation.

1.2.1 Script Specifications

Different scripts of the source and target languages is the first hurdle that transliteration systems should tackle. Script is representative of one or more writing systems, and is composed of symbols used to represent text. All of the symbols have a common characteristic which justifies their consideration as a distinct set. One script can be used for several different languages; for example, Latin script covers all of Western Europe, and Arabic script is used for Arabic, and some of non-Semitic languages written with the Arabic alphabet including Persian, Urdu, Pashto, Malay, and Balti. On the other hand, some written languages require multiple scripts; for example, Japanese is written in at least three scripts: the Hiragana

[1] http://en.wikipedia.org/wiki/Persian_language

and Katakana syllabaries and the Kanji ideographs imported from China. Computational processing of such different language scripts requires awareness of the symbols comprising the language; for example the ability to handle different character encodings.

Also, some scripts are written using separate characters, while others introduce intermediate forms for characters that occur in the middle of a word. For example, in Persian script some letters change their form based on their position in the word. The character "پ" [p][2], is written "پـ" [p] when it appears in the beginning of a word, "ـپـ" [p] in the middle, and "ـپ" [p] at the end; however, this rule is sometimes violated when "پ" is adjoined to special letters such as "ا" [ɒ] in "پاپ" /pɒp/.

Another important aspect of language script is its text directionality. Some languages are written right-to-left (RTL), and some are written left-to-right (LTR). For example, Persian which uses a modified Arabic script is RTL, whereas English script is LTR.

In general a transliteration system, which manipulates characters of the words, should be designed carefully to process scripts of the source and target languages considering all of the above mentioned specifications.

1.2.2 Missing Sounds

Different human languages have their own sound structure, and symbols of the language script correspond to these sounds. If there is a missing sound in the letters of a language, single sounds are represented using digraphs and trigraphs. For example, an English digraph "sh" corresponds to the sound [ʃ]. Cross-lingual sound translation — the function of transliteration — introduces new sounds to a target language, which the target language does not necessarily accommodate. That is, sounds cannot inevitably be pronounced the same way as in their original language after being imported to the target language. Such sounds are conventionally substituted by a sequence of the target language letters. For example, the sound of the Persian letter "خ" [x] has no equivalent in English. Persian speakers usually transliterate it to the digraph "kh" in English, whereas for some other languages with Latin script, such as Czech, it is written as "ch". Transliteration systems are expected to learn both the convention of

[2]According to the standards of phonetics and phonology, the phonetic representation of a sound is shown using [ ], and the phonemes are represented by / / (refer to Appendix B and C for a guide on pronunciations and further explanation).

writing the missing sounds in each of the languages involved in isolation, and the convention of exporting the sounds from one language to the other.

1.2.3 Transliteration Variants

The evaluation process of transliteration is not straightforward. Transliteration is a creative process that allows multiple variants of a source term be valid based on the opinions of different human transliterator. While gathering all possible variants for all the words in one corpus is not feasible — simply because not all speakers of those languages can be called upon in the evaluation process — there is no particular standard for such a comparison, other than conventions developed among nations. Further, new names of companies, products, and people are introduced on a daily basis, which leaves no way of having a standard transliteration. Therefore, evaluation of transliteration systems becomes problematic, in particular when comparing the performance of different systems.

1.3 Scope of the Thesis

This thesis focuses on the specific problem of machine transliteration for English and Persian, a previously unstudied language pair; it involves both English to Persian (English-Persian), and Persian to English (Persian-English) transliterations. Its main aim is to improve machine transliteration effectiveness for the language-pair English and Persian.

More specifically, we divide the problem into two parts: transliteration effectiveness and transliteration evaluation. The corresponding research problems investigated in this thesis and our contributions in solving them are explained in the following sub-sections.

1.3.1 Transliteration Effectiveness

The following problems are examined in regard to transliteration effectiveness.

1. *Investigation of existing methods for transliterating English and Persian.*

Many different transliteration systems are reported in the literature for a variety of language pairs. The effectiveness of such systems, defined based upon their accuracy in generating transliterations, is their most important aspect. Having English and Persian targeted for our transliteration task, we first investigate the effectiveness of existing transliteration

methods. We explore these approaches to determine which are the most effective for our transliteration task. We also investigate the opportunities of modifying these existing approaches to improve the effectiveness of English-Persian and Persian-English transliterations.

Since the Persian language was not previously studied in the literature for any transliteration task, we begin by exploring the widely used transliteration methods based on n-grams that have been proposed for other language-pairs. Those approaches that use n-grams define a specific context length for the characters under transliteration in each word. Then, we vary this context in the source word and examine its effect on the efficacy of the system. The impact of using a target language model is also explored, following the trend of using language models in the generated target word. Some of these methods, although very promising for some other languages, are not as effective for our Persian-English and English-Persian tasks.

2. *Determining language-specific features that assist English-Persian and Persian-English transliteration.*

Transliteration is very language specific. More precisely, it is language script and language user specific. We demonstrate this observation by investigating transliteration failures in the examined methods. Based on which, we propose the usage of shallow but effective habits of transliteration (that build on inter-lingual sound translation conventions), and script-specific features. Our novel methods are formed based on these features, specifically the consonant and vowel concepts of the characters in a word and their order. These approaches show significant improvements over the methods that use only n-grams.

3. *Exploration of the effect of character alignment on machine transliteration.*

Automatic transliteration techniques require an initial knowledge of transformation rules from a source language to a target language. One method of learning such rules uses character alignment of source and target words. Since such an alignment is used as part of the transliteration process, the accuracy of alignments can affect the accuracy of transliteration. Therefore, transliteration accuracy can be enhanced by improving the alignment process. We investigate the effects of a general MT alignment tool (GIZA++) used at the character level, and develope a novel alignment algorithm based on the consonant-vowel concepts in English-Persian transliteration. Our new approaches lead to significant improvements in

transliteration accuracy.

4. *Exploring system combination techniques for transliteration.*

In a further attempt to improve the accuracy of our transliteration systems, we explore the combination of the outputs from multiple systems. System combination has been successful in many different fields of language processing, such as parsing, part-of-speech tagging and named-entity recognition. Literature on machine transliteration also explores some methods of linear combination of systems known as hybridisation. We investigate possible combinations of transliteration systems in order to improve English-Persian and Persian-English transliteration efficacy. We consider new methods for the combinations of evidence in transliteration, exploring two core problems: specifying methods of system combination that lead to more accurate results, and indentifying which transliteration systems should participate in the combination. Our results show a significant increase in the accuracy of both English-Persian and Persian-English transliterations using our proposed combination of specific systems, demonstrating that careful system combination is a successful approach in improving transliteration effectiveness.

5. *Improving back-transliteration.*

Once a word is transliterated, it is a loan word in the second language. Transliterating the loan word back to its original form is called back-transliteration. Back-transliteration is usually more difficult than transliteration because some information is lost in the forward transliteration process; for example, missing sounds contribute in this information-loss. Back-transliteration is also stricter: we cannot have transliteration variants, and should re-produce exactly the same original word.

In this thesis, for backward transliteration, we start by examining how different transliteration methods proposed for forward transliteration perform in the backward transliteration task. We then investigate the modifications of these approaches to create a back-transliteration system which generates English words from their transliterated Persian word. We propose a novel method of source word segmentation in our transliteration approach to more effectively transliterate Persian words back to English.

1.3.2 Transliteration Evaluation

The following problems are investigated in regard to transliteration evaluation.

1. *Investigation of parameters affecting the comparison of transliteration systems.*

Machine transliteration between any two language pairs requires a bilingual corpus of source and target words for evaluation. In addition, those transliteration approaches that rely on automatic learning require such a corpus for their training. The number of valid transliteration variants included in a corpus might affect the comparison of performance of multiple systems. Since existence of multiple transliteration variants for each word is a sign of disagreement among humans, not accounting for this variety may bias the evaluations towards particular systems.

In this thesis, we explore the effect that altering corpora has on transliteration evaluation. Concerned about conditions of fair judgements over different systems, we study this phenomenon in regard to the corpus construction process, metrics of evaluation, and human transliterators.

2. *Constructing bilingual transliteration corpora.*

Persian has not been studied in any transliteration study before and there was no corpus available for our training and testing requirements. We therefore needed to build corpora for our experiments. Following such a corpus construction process, we investigate the effect of the differing origin of source words provided to transliterators on the perceived accuracy of the transliteration systems. We also design careful experiments to find the number of transliterators that should transliterate each of the source words to enable fair performance comparison between the systems. We determine the minimum size of a corpus in terms of the number of source words that a transliteration corpus should consist of for robust performance.

We report our findings on the construction of a corpus for transliteration evaluation. An important result of this study is that creating a transliteration corpus should be done with the aid of at least four transliterators. This guarantees that system rankings remain stable across different corpora. Our experiments also specify the minimum size of a corpus depending upon the origin of the words contributing the corpus. If source words come from multiple origins, the size of corpus should be higher than for single-origin words. Our conclusions

reported in this thesis highlight the effects that a corpus may have on the evaluation of system effectiveness.

1.4 Structure of the Thesis

This thesis is comprised of seven chapters:

Chapter 1 gives an introduction to the research problems which this thesis focuses on. It outlines the research questions addressed in this thesis.

Chapter 2 provides a literature review on both machine translation and machine transliteration. The background provided on machine translation is focused on statistical alignment concepts and techniques that are applied in an alignment tool that we used in part of our work (GIZA++). The machine transliteration literature reviews discriminative and generative transliteration. The generative transliteration research is divided into three main categories: phonetic-based, spelling-based and hybrid methods. The research methodology undertaken in this thesis is also explained in this chapter.

Chapter 3 illustrates the baseline systems that are adapted from approaches proposed in the literature for other languages. Transliteration methods based on n-grams, and application of target language models on English-Persian, and Persian-English transliteration are explored and reported. In particular, we investigate the effect of context size on transliteration accuracy.

Chapter 4 discusses novel approaches of transliteration proposed for the English and Persian language pair. A new character alignment method that is effective for English to Persian transliteration is also proposed. Furthermore, a novel method for back-transliteration is presented in this chapter.

Chapter 5 investigates methods of combining transliteration systems, and the effect of such integration on effectiveness of machine transliteration.

Chapter 6 contains the experiments and discussions of the effects that a training and testing corpus, and its construction process, have on transliteration effectiveness. Evaluation measures of transliteration accuracy are also closely studied.

Chapter 7 presents our conclusions and discusses further research that can be pursued based on the work reported in this thesis.

Appendix B explains the terminology used throughout this thesis.

Chapter 2

Background

"Those who cannot remember the past are condemned to repeat it."

— *George Santayana*

This background chapter is divided into three main sections: machine translation, machine transliteration, and methodology. The machine translation section gives a broad overview of the general approaches of machine translation, particularly word alignment techniques explored in automatic translation area. The main focus is alignment of words in parallel sentences using statistical approaches. The machine transliteration section presents previous work for both transliteration extraction and transliteration generation, techniques that are used in our transliteration approaches in Chapters 3 to 5. In the transliteration generation section — which is the main subject of this thesis — literature on the three main categories of phonetic-based, spelling-based and hybrid methods is reviewed. Finally, the methodology section covers information on bilingual transliteration corpora, and techniques that are used in the experiments reported in the following chapters.

2.1 Machine Translation

Machine translation (MT) is the automatic translation of texts written in one natural language (known as the *source* language) to another natural language (known as the *target* language). MT is studied as part of computational linguistics, and involves natural language

understanding and language processing. Machine translation is an important field to study because of its scientific, social, political, and commercial applications. For example, machine translation can lead to huge savings for governments. The European Union spends more than one billion dollars on translation costs per year, which could be reduced using MT applications. Machine translation also has many academic applications; for instance, linking between languages by for example knowing the translations of words, phrases, or sentences allows for transferring resources from one language to another.

The idea of automatic translation was proposed by Warren Weaver (1947). He considered the translation problem as a decoding task and proposed the use of cryptographic methods, as stated in his famous quote:

> "I have a text in front of me which is written in Russian but I am going to pretend that it is really written in English and that it has been coded in some strange symbols. All I need to do is strip off the code in order to retrieve the information contained in the text."

Although initial efforts of scientists in the Rockefeller research group to fulfil this idea was abandoned at the time due to computational complexity, it was later followed as a valuable research topic in computational linguistics [Brown et al., 1990].

A basic MT system uses a bilingual dictionary to substitute the words of a text in one language with its equivalents in the other, ignoring any linguistic feature of both languages. A more complicated system deals with more realistic features such as language grammar and the role of words in the context; in other words, it deals with semantic and pragmatic information. Major challenges that such systems face are word sense disambiguation and named entities.

Word sense disambiguation involves the association of a meaning (sense) to a given word in a text that represents the most appropriate sense in that context [Ide and Véronis, 1998]. For example, the word "cold" has several senses. It may refer to a disease, a temperature sensation, or a natural phenomenon. The context in which an instance of the ambiguous word appears determines the specific sense intended. For example, in "She is taking medicine for her cold" the disease sense is intended, while in "Let's go home, I feel cold" the temperature sense is meant [Humphrey et al., 2006]. While detemining the right sense of a word during

translation can be hard even for a human who understands the meanings from the context, it is a challenging task for machine [Vickrey et al., 2005].

Named entities are words or sequences of words that represent the name of a person, organization, location, date, time, quantities, or technical terms. Named entity recognition has been studied to tag such words or phrases for many applications such as text summarisation, information retrieval, speech recognition, and machine translation. Since such terms usually do not appear in dictionaries, most named entities are subject to transliteration when it comes to translating the text that contains such terms. Some semi-automatic machine translation systems, however, expect their user to provide them with the list of proper names in the text and their translations in order to be accurate in their output.

In general, machine translation paradigms are divided into four categories [Arnold et al., 2001]:

1. Dictionary-based machine translation

   In this approach, the main translation source is a bilingual dictionary which in fact defines the strength of the translation by the number of successful dictionary lookups. This method normally overlooks the relationship between source words.

2. Statistical machine translation

   Statistical machine translation relies on bilingual corpora to train the MT system. It is effective if similar texts have been seen in the training corpora. The main problem with this method is that such corpora are not readily available.

3. Example-based machine translation

   Example-based machine translation uses a knowledge-base and a bilingual corpus to perform case-based reasoning to map the source to the target.

4. Interlingua machine translation

   Interlingua machine translation is a rule-based model transforming the source language text to an intermediate language text, and then from the intermediate language to the target language.

In recent years, statistical machine translation (SMT) has been the most popular approach [Koehn, 2004; Pang et al., 2005]. We provide details on SMT techniques and related

concepts in this chapter. Most machine transliteration techniques are based upon SMT ideas, which motivates us to explain it in more detail.

2.1.1 Text Corpus

In principle, any collection of two or more texts can be called a corpus. In linguistics, a corpus — or text corpus — is a large and structured set of texts. A corpus in its modern linguistics definition has four main characteristics [McEnery and Wilson, 1996].

1. Sampling and representativeness. Usually, an individual text or author is not of interest in linguistics, but a whole variety of language is often studied. That means a corpus should not be biased towards specific authors, and representative of the variety in the language under examination. In other words, a corpus should provide us with a picture of the tendencies of that variety, as well as their proportions.

2. Finite size

3. Machine-readable
   A machine-readable corpus is more advantageous than written or spoken formats. A text machine-readable corpus can be searched and manipulated at speed, and it can easily be enriched with extra information.

4. Standard reference
   A corpus that is widely available to other researchers is a standard reference for the language variety it represents.

A corpus may contain texts in a single language (*monolingual*), two languages (*bilingual* or *bitexts*), or multiple languages (*multilingual*). Bilingual and multilingual corpora can be parallel or comparable (non-parallel). There are some disagreements in the literature in definitions of these two types of corpora [Pearson, 1998]. In this thesis, we use the following definitions that have been widely used in the recent literature.

Definition 1 *A parallel corpus is a collection of texts in two or more languages. Each of the texts is an exact translation of one or more other languages, and the direction of the translation can be unknown.*

Parallel corpora are attractive to researchers because of the opportunity of aligning translated texts (at the sentence, word, or even tag level), and because they can give insights into the nature of translation. However, the strict property of parallel corpora in providing exact translations makes them hard to construct or obtain. Comparable corpora, as described in Definition 2, are another popular resource for computational lexicography and machine translation.

Definition 2 *A comparable corpus is a collection of texts in two or more languages. The texts are similar, meaning that they contain similar information but they are not exact translation of each other. No information is available regarding the similarity.*

Corpora, in general, provide the main knowledge-base for corpus linguistics. *Corpus linguistics* is the study of language as expressed in its samples (in the form of a corpus), or real world text, to represent an approach to deriving a set of abstract rules by which a natural language is governed or relates to another language. The analysis and processing of various types of corpora is the subject of much work in computational linguistics, speech recognition and machine translation [McEnery and Wilson, 1996].

One of the recent challenges for corpus linguistics is using the World Wide Web as Corpus. Investigation of methods for culling data from the Web has introduced two approaches in corpus linguistics: "Web-as-corpus" and "Web-for-corpus-building" [Hundt, 2006]. We discuss this phenomenon in section 2.4.

2.1.2 Statistical Machine Translation

Statistical machine translation (SMT) is an approach of automatic translation that uses statistical information from parallel corpus (Definition 1). All other machine translation methods focus on the steps of translation, whereas SMT focuses only on the output and not the process [Jurafsky and Martin, 2008].

Statistical MT, in general, follows a sequence of modelling, parameter estimation, and search, where each of these are specified based on the complexity of the SMT model (word-, phrase-, or syntax-based). An SMT model can be considered as a function of faithfulness to the source language, and fluency in the target language [Jurafsky and Martin, 2008]. With the intention of maximising both of these two parameters, the fundamental model of SMT is

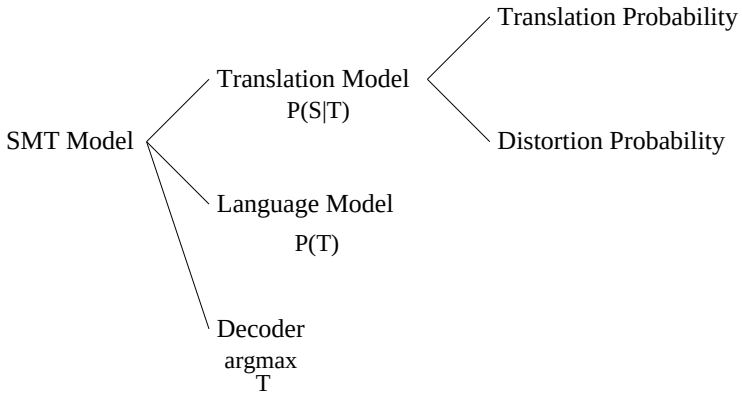


Figure 2.1: A general diagram of components of a statistical machine translation system.

generally defined based on a *translation model* (faithfulness) and a *language model* (fluency) as

$$P(S,T) = argmax_T P(S|T)P(T), \tag{2.1}$$

where S is a sentence in the source language, T is a sentence in the target language, $P(S|T)$ represents translation model, and $P(T)$ denotes target language model. The translation model represents the probability of T being a translation of S, and language model indicates the probability of having a string in the target language with the word-order generated in T. An overview of a SMT system and its components is shown in Figure 2.1; in the rest of this section, we describe each of these components.

The SMT model (Equation 2.1) is originally formed using Bayesian noisy channel theory [Brown et al., 1990]. The assumption is that sentences in the target language are in fact in the source language but corrupted by noise. Therefore, we need a decoder that, given the sentence S, produces the most probable sentence T. The target model or $P(T)$ specifies sentences that are valid in the target language, and channel model or $P(S|T)$ explains the influence of noise that changed the source sentence from S to T [Brown et al., 1990; Knight, 1999].

The translation model is dependant on two probabilities: translation and distortion. The *translation probability* can be shown as $\tau(s_i|t_j)$ where t_j is a word in the target sentence $T = t_1, t_2, \ldots, t_J$, and s_i is a word in the source sentence $S = s_1, s_2, \ldots, s_I$. The factor

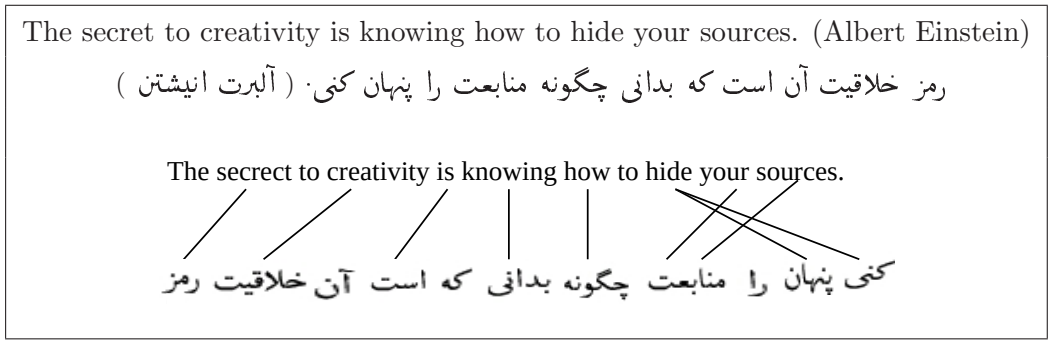


Figure 2.2: A sample English-Persian translated sentence-pair. The correspondence of words between the sentences is shown in graphical representation (with Persian words written left to right for clarity).

$\tau(s_i|t_j)$ represents the probability of translating s_i to t_j.

Distortion relates to re-ordering of words in the translation process. It measures the distance between the source word and its corresponding translated word(s) [Brown et al., 1993]. To clarify, consider Figure 2.2 which shows an English-Persian pair of sentences and the correspondence of words in the sentences. Word-by-word translation of the Persian sentence is "secret creativity {that} is {that} know{you} how sources-your {a preposition[1]} hide{you}", with { } representing words that are added in translation process for fluency in the target language. It can be seen that the word order has changed translating English to Persian. For example, the word "how" in position 6 is translated to the word "چگونه" /tʃeguːne/ in position 7. *Distortion probability* is the probability of these words to be translated with the same position difference in other texts. Many different methods are proposed in the literature for handling distortion probability. A basic distortion scheme relies on the absolute position of correspondent words in a sentence-pair [Brown et al., 1993]. In our previous example, the distortion probability can be shown as $d(7|6)$, which is the probability of an English word in position 6 generates a Persian word in position 7. A similar scheme takes the length of sentences into account. That is, in the previous example, the probability is conditioned on four parameters as $d(7|6, 11, 11)$; which translates to probability of given an English sentence of 11 words length, the word in position 6 generates a translation in position 7 of a Persian

[1]The Persian preposition "را" /rɒ/ has no equivalent in English.

sentence of length 11. Some other methods look at the relative distance instead of absolute distance [Brown et al., 1993; Jurafsky and Martin, 2008]. A phrase-based SMT model used by Koehn et al. [2003] considers the span between the phrases in the target sentence that are translations of consecutive source phrases. Let $\hat{s}_{i-1}$ and $\hat{s}_i$ be two phrases in the source language, and $\hat{t}_{i-1}$ and $\hat{t}_i$ their corresponding target phrases. Koehn et al. [2003] parameterise the distortion function as $d(b_i - a_{i-1})$, where a_{i-1} is the end position of target phrase $\hat{t}_{i-1}$, and b_i is the start position of the target phrase $\hat{t}_i$. To penalise larger distances between consecutive phrases, such distortion function can be modelled as $d(b_i - a_{i-1}) = \alpha^{|b_i - a_{i-1} - 1|}$.

To be complete, the MT system explained so far needs a model of training and a model of decoding. When training the model, parameters of the SMT model are learnt (for example the α parameter for distortion). In decoding model, the system finds the hidden sentence T from the noisy sentence S [Jurafsky and Martin, 2008]. These two models are demonstrated below.

Training Model

We introduced the translation model and the language model as the main components of a SMT system. These two models require a training stage to compute their probabilities. We therefore need to specify resources on which these probabilities are learnt, and techniques of learning. In terms of resources, a language model needs a large monolingual corpus in the target language for training, and its accuracy is dependent on size and quality of this corpus. The translation model on the other hand, requires large bilingual parallel corpora which is harder to obtain in comparison to monolingual corpus [Arnold et al., 2001].

The translation model $P(S|T)$ depends on the translation and distortion probabilities. Depending on the translation method (word- or phrase-based) and the distortion scheme used, the methods of computing of translation model can be different. For example, Koehn et al. [2003] in their phrase-based SMT model used the translation model

$$P(S|T) = \prod_{i=1}^{I} \tau(\hat{s}_i|\hat{t}_i).d(b_i - a_{i-1}),$$

where the distortion probability d is as above, and translation probability

$$\tau(\hat{s}|\hat{t}) = \frac{\text{freq}(\hat{t}, \hat{s})}{\sum_{\hat{t}} \text{freq}(\hat{t}, \hat{s})}.$$

where $\text{freq}(\hat{t}, \hat{s})$ indicates the frequency thta $\hat{s}$ is translated to $\hat{t}$ in the corpus.

The translation and distortion probabilities are achievable from a sentence-aligned bilingual corpus. If the relation of the source and target phrases were known in these sentences (similar to the one represented graphically in Figure 2.2 on page 18), then translation model probabilities would be easy to obtain. However, such corpora with aligned words and phrases do not exist. Word-alignment techniques are used to discover the relationships between the words in aligned sentences. Translation tables can be constructed using aligned sentences, and if a phrase-based model is developed, phrases can be extracted. We detail the word alignment approaches in Section 2.1.3.

The language model $P(T)$ determines whether the sentences highly scored by translation model are fluent, or grammatically correct in the target language. A translation model only takes care of sentence components, such as words or phrases, and not the sentence overall. For example, according to the translation model "the secret to creativity is knowing how to hide your resources" can be as probable as "to creativity the secret your resources to hide how is knowing". Hence, a language model with the assistance of a large corpus of texts in target language should penalise the second scrambled sentence. That is, if trained well on a large enough English corpus, a language model ranks the first sentence higher than the second. However, language models themselves work on fixed length word orders called n-grams (sequences of n words). If the pattern in the sentence is not seen in the training corpus, even a good sentence may get a low score [Knight, 1999]. Therefore, smoothing techniques are required to cover for unseen sequences [Chen and Goodman, 1996].

Decoding Model

Once the MT system is trained at the word or phrase level, it can be used to translate new sentences. Once more, considering the main SMT model in Equation 2.1 on page 17, we expect to choose a sentence T as a translation of the sentence S so that it maximises both the translation model and language model probabilities. In other words, it must be faithful to the origin text as much as possible, as well as fluent in the target language [Jurafsky and

Martin, 2008]. Such maximisation is called decoding. Since the decoder should find the ideal sentence which maximises the Equation 2.1 among all the probable sentences, decoding is a search problem.

MT decoders are a kind of heuristic or informed search; mainly from a category of best-first-search called stack decoding [Jelinek, 1969]. SMT systems that are based on IBM Models [Brown et al., 1996] use stack decoding for word-based machine translation. Recently, phrase-based SMT uses beam search rather than best-first search, particularly after the release of a publicly available SMT decoder called Pharaoh [Koehn, 2004].

In general, such decoders search through all possible sentences that can be generated from the source sentence (not all the English sentences available) and assign each a cost. The cost is based on the translation model and language model probabilities of sentences under generation. In Pharaoh the cost function is

$$\text{cost}(S,T) = \prod_{i \in L} \tau(\hat{s}_i|\hat{t}_i).d(b_i - a_{i-1})P(T),$$

where $L = (S,T)$ is a partial translation meaning that it can be a phrase or part of a sentence. This cost can be a combination of current cost and future cost. Future cost is an estimate of cost of remaining phrases to be translated [Jurafsky and Martin, 2008]. The candidate translation should cover all the words in the source sentence as well as minimising the cost. This cost can be computed using the Viterbi algorithm [Viterbi, 1967]. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states, called the Viterbi path, which specifies a sequence of observed events, particularly in the context of Hidden Markov Models.

In recent literature [Chiang, 2005; Hassan et al., 2007], short sentences are penalised and therefore the main SMT model in Equation 2.1 is changed to

$$P(S,T) = argmax_T P(S|T)P(T)\omega(|T|),$$

where $\omega(|T|)$ is a function that rewards longer sentences that were unfairly punished by language model $P(T)$ [Jurafsky and Martin, 2008]; for example, word penalty can be $\omega(x) = exp(-\lambda|x|)$ for some $\lambda > 0$ [Chiang, 2005].

Many other approaches are suggested in the literature for solving the decoding problem in MT, which are outside the scope of this thesis; however they all share the same concepts as explained above.

2.1.3 Word Alignment

Word alignment is an important component of any SMT system. Research in development of alignment algorithms was pioneered by the IBM Watson research group by developing five IBM Models [Brown et al., 1993]. In this section, we provide a summary of IBM Model 1, Model 3 [Brown et al., 1993], and the hidden Markov model (HMM) [Vogel et al., 1996; Toutanova et al., 2002; Och and Ney, 2003] to give an overview of how alignment in performed in a machine translation system.

Word-alignment is a mapping between the words of a pair of sentences that are translation of each other. IBM alignment models assume that alignment is restricted to two sentences and does not propagate across sentence boundaries. An example of such a mapping is shown in Figure 2.2 on page 18. As can be seen in the figure, there are some words in the source English sentence that have no equivalent in Persian, and there are some target Persian words that are not specifically generated from any of the source words. Such phenomena can be modelled by considering NULL words in the source sentence, which are counted as sources of *spurious words* in the target sentence [Knight, 1999]. Depending on the algorithm, alignments can be one-to-one, one-to-many, or many-to-many.

In general, the relationship of the alignment a and a pair of sentences S and T can be modelled as

$$P(S|T) = \sum_{a} P(S, a|T). \tag{2.2}$$

Equation 2.2 is called a generative probability model [Knight, 1999; Jurafsky and Martin, 2008].

All of the alignment models use the concepts of training and decoding to find the best alignment between words of S and T. A brief summary of the IBM Models and the HMM model is given below[2].

[2]Alignment models are mainly explained following the "Machine Translation" chapter of Jurafsky and Martin [2008] (chapter 25), unless otherwise cited.

IBM Model 1

IBM Model 1 is the simplest alignment model used on parallel bilingual corpora. It was mainly developed to provide initial estimates for more complex word-alignment methods [Moore, 2004]. Model 1 is a generative story of how a target sentence T is generated from a source sentence $S = s_1 s_2 \dots s_I$. It has three steps to calculate the generative probability model in Equation 2.2:

1. Choose a length J for the target sentence.
2. Choose an alignment $a = a_1 a_2 \dots a_J$ between the source and target sentences.
3. For each position j is the alignment translate the source word — connected to this position through alignment — to a target word t_j.

If t_{a_j} is the target word aligned to the source word s_j, and $\tau(s_j|t_{a_j})$ represents the probability of translating s_j to t_{a_j} — with the assumption of knowing a and T in advance — then the probability of generating sentence S (step 3) is

$$P(S|T,a) = \prod_{j=1}^{J} \tau(s_j|t_{a_j}). \tag{2.3}$$

As explained previously, some words that appear in the target sentence are spurious. That is, no source word has directly generated them. To generate such words, the IBM models consider a NULL word at the beginning of the source sentence that is responsible for generating spurious words in T.

IBM Model 1 has two simplifying assumptions: first, all alignments are equally likely, and second, the probability of choosing length J for a is a small constant ϵ. Given a source sentence of length I and a target sentence of length J, the number of possible alignments between these two sentences is $(I+1)^J$ (adding a NULL at the beginning of S). Therefore, the probability of choosing any of the possible alignments is:

$$P(a|T) = \frac{\epsilon}{(I+1)^J}. \tag{2.4}$$

Combining Equations 2.3 and 2.4, the probability of generating T via a specific alignment can be calculated:

$$P(S, a|T) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} \tau(s_j|t_{a_j}). \tag{2.5}$$

The generative probability model of IBM Model 1 assigns a probability to each possible target sentence; therefore, to calculate $P(T|S)$, we sum over all possible alignments,

$$P(S|T) = \sum_{a} \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} \tau(s_j|t_{a_j}). \tag{2.6}$$

Finally, to find the best alignment for a sentence-pair (S, T), decoding is required. The Viterbi algorithm finds the best alignment, where the alignment of each word is independent from the best alignment of its surrounding words. Therefore, the best alignment is

$$\begin{aligned} \hat{a} &= argmax_a P(S, a|T) \\ &= argmax_a \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} \tau(s_j|t_{a_j}) \\ &= argmax_a \prod_{j=1}^{J} \tau(s_j|t_{a_j}). \end{aligned}$$

The parameters of IBM Model 1 for any pair of languages are estimated using EM (estimation-maximisation) algorithm [Dempster et al., 1977]. EM algorithm finds maximum likelihood estimates of parameters in a probabilistic model, where the model depends on unobserved latent variables. EM alternates between performing an estimation (E) step, which estimates the likelihood by including the latent variables pretending that they are observed, and a maximization (M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found in the E step. The parameters found in the M step are then used to begin another E step, and the process is repeated till the optimum parameters values are found [Dempster et al., 1977].

Training on a sentence-aligned parallel corpus normally starts with uniform distribution of all translation probabilities over the target language vocabulary [Moore, 2004]. More specifically, translation probabilities for Model 1 are computed in three steps: first, for each source word s all target words t that co-occur at least once with s are collected; second,

$\tau(s|t)$ probabilities are uniformly initialised, for example using co-occurrence information $\tau(s|t) = \frac{1}{\text{number of co-occurrences of } s \text{ and } t}$; third, translation probabilities are iteratively refined.

Despite the fact that Model 1 is simple and widely used, there are some limitations in this model. One-to-many and many-to-one alignments are not supported and each target word should be generated by at most one source word. No distortion is implemented in this model. The position of corresponding source and target words are independent, and therefore, phrases cannot be modelled. Also, the fact that some source words generate multiple target words is ignored [Moore, 2004] (this concept is called fertility and explained later in Model 3). Some of these problems are addressed in other IBM models, and some in improved algorithms investigated by other researchers.

IBM Model 3

Translation from one language to the other is a creative task. There are some words in a language that generate multiple words in a foreign language, there are words that are just omitted in the translation task, and words that appear in the target sentence, only for fluency. In the translation example shown in Figure 2.2, there is an English word that generated two Persian words: "hide" is translated to "پنهان" /penhɒn/ and "کنی" /koniː/. The formal way to express this phenomenon is that the fertility of the word "hide" is 2. *Fertility* is the number of words (zero or more) that a source word generates at translation time in a target language [Arnold et al., 2001]. IBM Models 3, 4, and 5 are fertility-based alignments, which means they support one-to-many alignment.

The generative story of Model 3 has five main steps [Knight, 1999; Jurafsky and Martin, 2008]:

1. For each source word s_i in the sentence $S = s_1 s_2 \dots s_I$, choose a fertility ϕ_i with probability $n(\phi_i|s_i)$.

2. For the NULL word added at the beginning of the source sentence, choose a fertility ϕ_0. NULL is responsible for spurious words. The spurious word generation is different from Model 1, where these words are continuously generated after each real word.

3. By determining the fertility of each source word, the number of target words is known. Generate target words based on their corresponding source words. Translation is only

dependant on the source word.

4. For each non-spurious word choose a position using the distortion probability $d(j|a_j, I, J)$.

5. For each of the spurious words, choose a vacant position based on a total probability of $\frac{1}{\phi_0!}$.

Model 3 has four additional parameters in comparison to Model 1: ϕ, n, d, and p_1. The fertility of each word is given by $n(\phi_i|s_i)$. For example, $n(2|\text{hide})$ indicates the fertility of 2 for the English word "hide" translating to Persian. Distortion d in Model 3 uses positions of the source and target words, in addition to the length of the source and target sentences. Similar to Model 1, $\tau(s_i|t_j)$ denotes the probability of generating t_j from the source word s_i.

Since fertility ϕ_i only specifies non-spurious words, a p_1 probability is assigned for generating spurious words. For each real word, a spurious word is generated with probability of p_1.

To compute $P(S|T)$, first $P(S, a|T)$ should be calculated and then summed over the possible alignments (Equation 2.2). $P(S, a|T)$ consists of two components, one for generating the real words, and the other for the spurious words. Real words are calculated as

$$P_{\text{real}} = \prod_{i=1}^{I} n(\phi_i|s_i) \times \prod_{j=1}^{J} \tau(t_j|s_{a_j}) \times \prod_{j=1}^{J} d(j|a_j, I, J),$$

where

$$J = \sum_{i=1}^{J} \phi_i,$$

which is composed of fertility, distortion and translation probabilities. The cost of generating spurious words is divided into three parts of spurious generation, insertion, and permutations of these words into their final target positions.

$$P_{\text{spurious}} = \binom{J - \phi_0}{\phi_0} p_0^{J-2\phi_0} p_1^{\phi_0} \times \frac{1}{\phi_0!} \times \prod_{i=0}^{I} \phi_0!$$

Hence, the total probability of $P(S, a|T)$ is calculated as

$$P(S, a|T) = P_{\text{real}} \times P_{\text{spurious}} \tag{2.7}$$

Training in Model 3 involves learning all of the introduced parameters. Alignments and probability models are learnt using the EM algorithm. In Model 3, unlike Model 1, all possible alignments should be computed. That is, translation probabilities cannot be computed independently of each other and IBM Model 3 should work with all the $(I+1)^J$ possible alignments. Such a calculation is usually computationally infeasible. Thus, using an iterative or bootstrapping method, the number of alignments should be reduced. Best alignments are normally computed with Model 1, and some of the alignments are changed to generate a set of likely alignments. This process is called *pegging* [Germann, 2001; Jurafsky and Martin, 2008].

The advantage of Model 3 over the previous models (Model 1 and 2) is that fertility and permutation of the words are considered. Model 3, however, is deficient in that it wastes some probabilities by assigning multiple words to one position and keeping other positions vacant. This problem is addressed in IBM Model 5 [Brown et al., 1993; Jurafsky and Martin, 2008] which we do not describe here.

HMM Alignment

HMM word-alignment was proposed to address some of the limitations in the IBM models. As discussed previously, one of the simplifying assumptions of IBM Model 1 is that all the $(I+1)^J$ possible alignments are equally likely. In practice however, words that are in a neighbourhood tend to be aligned together. In the example shown in Figure 2.2 on page 18, we can see that alignments keep *locality* by tending to align to their neighbour target words. This is more prominent in some language-pairs, for example English and French. Locality is implemented in the HMM alignment model where alignment of each word depends on the alignment of its preceding words. HMM restructures the $P(S,a|T)$ probability in IBM Model 1 using the chain rule

$$\begin{aligned} P(t_1^J, a_1^J|s_1^I) &= P(J|s_1^I) \times \prod_{j=1}^{J} P(t_j, a_j|t_1^{j-1}, a_1^{j-1}, s_1^I) \\ &= P(J|s_1^I) \times \prod_{j=1}^{J} P(t_j|t_1^{j-1}, a_1^{j-1}, s_1^I) \times P(a_j|t_1^{j-1}, a_1^{j-1}, s_1^I). \end{aligned} \tag{2.8}$$

According to Equation 2.8, $P(S, a|T)$ is made of three probabilities: an alignment probability $P(a_j|t_1^{j-1}, a_1^{j-1}, s_1^I)$, a lexicon probability $P(t_j|t_1^{j-1}, a_1^{j-1}, s_1^I)$, and a length probability $P(J|s_1^I)$. Standard Markov simplification assumptions can be applied to this equation for easier calculations. We can assume that the alignment of each word is only dependent on the previous word. Also, the probability of generating a target word t_j is only dependant on the source word s_{a_j} at position a_j (aligned to t_j). Therefore, we can re-write Equation 2.8 as

$$P(t_1^J, a_1^J|s_1^I) = P(J|I) \times \prod_{j=1}^{J} P(a_j|a_{j-1}, I)P(t_j|s_{a_j}). \tag{2.9}$$

To calculate $P(t_1^J|s_1^I)$, the $P(t_1^J, a_1^J|s_1^I)$ probability should be summed over the alignments:

$$P(t_1^J|s_1^I) = P(J|I) \times \sum_a \prod_{j=1}^{J} P(a_j|a_{j-1}, I)P(t_j|s_{a_j}). \tag{2.10}$$

In Equation 2.10, all positions are absolute. That is, although locality of the alignments is captured using $P(a_j|a_{j-1}, I)$, distances are absolute. In order to make them relative, a *jump width* is defined. Jump width is the distance between the positions of two words. That is, if i and i' stand for two positions in the source sentence, the jump width is $i - i'$. A non-negative function of jump width is defined as

$$P(i|i', I) = \frac{c(i - i')}{\sum_{i''=1}^{I} c(i'' - i')}.$$

Other modifications on the basic HMM model alignment has also been proposed. For example, Toutanova et al. [2002] incorporated part-of-speech tags (word class) information to alignment. They also changed the NULL word alignments to prevent the problems of IBM Models in aligning everything to NULLs [Och and Ney, 2000].

Training and decoding for HMM model are done using the well-known Baum-Welch [Baum et al., 1970] and Viterbi algorithms.

In summary, IBM Model 1 only uses lexical probabilities to perform alignment. It is computationally inexpensive, but alignments are one-to-one only. Model 2 is similar to Model 1, but absolute positions are included. HMM uses lexical probabilities and relative positions, while IBM Model 3 introduces fertility probabilities. The distinguishing feature of

IBM Model 4 is inverted relative position alignment, and Model 5 is a non-deficient version of Model 4.

IBM Models have some limitations, including: only supporting one-to-many mappings; fertility-zero words are difficult for decoding; almost no syntactic information or word classes is considered; and long-distance word movement, and fluency of the output entirely depends on the target language model. Many improvements have been reported over these models. However, the next generation of word-based translation systems are phrase-based systems that solve some of these problems. Phrase translation captures context, and local word-reordering is incorporated. However, phrases are not integrated into the alignment model and they are extracted from word-aligned sentences.

The machine translation concepts explained here, particularly word alignment algorithms, are largely applied in machine transliteration at the character level. Specifically, word alignment and character alignment, and phrase mapping and substring mapping, resemble similar concepts in machine translation and transliteration, as we explain in the following sections.

A Word-Alignment Tool: GIZA++

GIZA++ is a SMT toolkit freely available for research purposes. The original program called *GIZA* was part of the SMT toolkit EGYPT, developed at the centre of language and speech processing at Johns Hopkins University by Al-Onaizan et al. [1999] during a summer workshop. The extended version of this toolkit is called GIZA++ and was developed by Och and Ney [2003]. It extends IBM Models 3, 4 and 5, alignment models using word classes, and includes: a HMM alignment model; more smoothing techniques for fertility, distortion, or alignment parameters; more efficient training of the fertility models; and more efficient pegging.

Word-by-word alignment is a sub-task of machine translation. Many experiments on machine translation use GIZA++ as their underlying alignment system. Machine transliteration systems have also benefited from such alignment, performing it at the character level [AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003b; Gao et al., 2004b].

GIZA++ inputs aligned sentences and outputs their aligned words. Each sentence-pair is stored in three lines. The first line is the number of times this sentence-pair has occurred in the parallel corpus. The second line represents the source sentence where each word is

Source: Persian

Target: English

Sentence pair (1548) source length 5 target length 4 alignment score : 0.00097478

u d i t

NULL ({ }) ا ({ }) و ({ 1 }) د ({ 2 }) ی ({ 3 }) ت ({ 4 })

Sentence pair (1549) source length 5 target length 6 alignment score : 8.55247e-07

u l l m a n

NULL ({ 5 }) ا ({ }) و ({ 1 }) ل ({ 2 3 }) م ({ 4 }) ن ({ 6 })

Source: English

Target: Persian

Sentence pair (1554) source length 4 target length 5 alignment score : 0.000372838

ت ی د و ا

NULL ({ 1 }) u ({ 2 }) d ({ 3 }) i ({ 4 }) t ({ 5 })

Sentence pair (1555) source length 6 target length 5 alignment score : 3.40974e-06

ن م ل و ا

NULL ({ 1 }) u ({ 2 }) l ({ }) l ({ 3 }) m ({ 4 }) a ({ }) n ({ 5 })

Figure 2.3: A GIZA++ *character-alignment sample with different source and target languages. Numbers in braces indicate the source character with which the target character to the left is aligned.*

replaced by a unique integer identifier from the vocabulary file, and the third is the target sentence in the same format.

A sample of character alignment done by GIZA++ for two transliterated words in English and Persian is shown in Figure 2.3. The output includes source and target aligned characters. Alignment is shown as a vector of numbers placed in front of each target character. As was done for NULL word added in the translation task, those characters that are spurious are assigned to NULL.

2.2 Machine Transliteration

The most conventional tool that aids translation is a dictionary. Typical dictionaries contain 50,000 to 150,000 entries. In practice however, many more words can be found in texts; for example, a collection of Associated Press newswire text collected over 10 months has 44 million words comprising 300,000 distinct words; with the difference to dictionaries being names, such as companies, people, places and products [Dale, 2007]. In such cases transliteration must occur, where the out-of-dictionary words are spelled out in the target language.

Literature on transliteration falls into two major groups: generative transliteration and transliteration extraction. Generative transliteration focuses on algorithms of transliterating newly appearing terms that do not exist in any translation lexicon. Transliteration extraction, on the other hand, enriches the translation lexicon by using existing transliteration instances on the Web, or other multilingual resources, to reduce the requirement of on-the-fly transliteration. We review both of these categories in this chapter; however, since the focus of this thesis is on generative transliteration techniques, the literature review emphasises generative transliteration, providing precise details on current approaches.

To make the explanations consistent in both the literature review (in particular generative transliteration approaches), and later illustrations of the methods developed in this thesis, a uniform formulation for transliteration models and the systems that follow these models is introduced below.

2.2.1 Transliteration Process

The generative transliteration process usually consists of a training stage, running once on a bilingual training corpus $B = \{(S, T)\}$, and a transliteration generation stage that produces target words T for each source word S, as shown in Figure 2.4. The training stage itself is composed of three tasks: alignment of source-target words or their sounds; segmentation (for example using graphemes or phonemes); and transformation rule generation. The transliteration stage consists of the segmentation of the test source word, and target word generation. Although this general process does not completely match with all the existing approaches, it is valid particularly for most recent methods. We explain the differences between these methods by comparing them with this general outline.

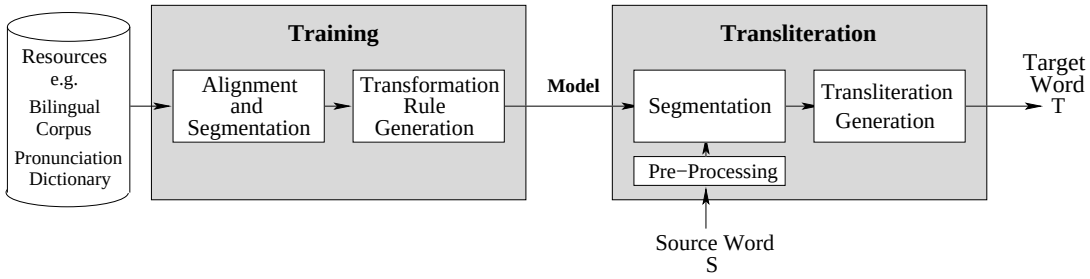


Figure 2.4: A general transliteration system.

2.2.2 Transliteration Model

Transliteration transforms words from a source language to a target language. In general, such transformations are performed character by character, or substring by substring (were words are segmented using grapheme or phoneme boundaries). It therefore requires particular rules to be followed.

Definition 3 *A transformation rule is denoted as* $\hat{S} \rightarrow (\hat{T}, p)$*, where* $\hat{S}$ *is a substring of the source alphabet,* $\hat{T}$ *is a substring of the target alphabet, and* p *is the probability of transliterating* $\hat{S}$ *to* $\hat{T}$*. For any* $\hat{S}$*,* $\hat{S} \rightarrow (\hat{T}_k, p_k) : \sum_{k=1}^{n} p_k = 1$*, assuming it appears as the head of* n *rules.*

We define a *transliteration model* as a method of forming transformation rules from training data. That is, patterns for segmenting source and target words, and possible incorporated context knowledge applied on specific training data, define a transliteration model. Such models form the core of a *transliteration system* that, given a source word as input, generates a ranked list of possible transliterations as output. More formally, we define a transliteration system as follows.

Definition 4 *A transliteration system* M *takes a source word* S *and outputs a ranked list* L*, with* (T_j, pr_j) *tuples as its elements. In each tuple,* T_j *is the* j^{th} *transliteration of the source word* S *generated with the* j^{th} *highest probability of* pr_j*.*

2.2.3 Bilingual Transliteration Corpus

Training and evaluation of transliteration systems require a bilingual corpus of source words and their transliterations (Definition 5). The number of acceptable transliterations of each source word can be more than one, therefore, in the corpus specifications we define T as a set of *transliteration variants* available in the corpus for a given source word S.

Definition 5 *A bilingual corpus B is the set $\{(S,T)\}$ of transliteration pairs, where $S = s_1..s_\ell$, $T = \{T_k\}$, and $T_k = t_1..t_m$; s_i is a letter in the source language alphabet, and t_j is a letter in the target language alphabet.*

Such a corpus, however, is not readily available for transliteration studies, particularly for languages with few computerised resources such as Persian. Section 2.5 on page 58 discusses how corpora were formed for the work in this thesis.

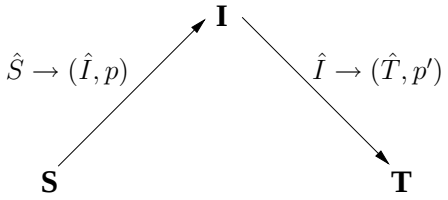
2.2.4 Evaluation Metrics

Typical evaluation measures for machine transliteration are word accuracy and character accuracy. Word accuracy — also referred as precision — measures the percentage of the words or phrases that are correctly transliterated, comparing the machine results with the available transliterations in a bilingual corpus (Definition 5). Character accuracy, being less strict than word accuracy, counts the characters that are correctly transliterated in each source word. Details of how these metrics are calculated are given in Section 2.5.4, page 60.

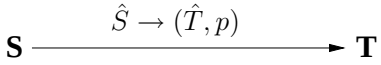
2.3 Approaches of Generative Transliteration

Generative transliteration is the process of transliterating a given term (word or phrase) from a source language to a target language. Many different generative transliteration methods have been studied in the literature, each of which introduces different methodologies or supports different languages. Due to the many varying attributes of these methods such as direction of transliteration, scripts of different languages, or different information sources, categorisation of these studies is not straightforward.

In terms of direction of transliteration, *forward* and *backward* transliteration is introduced. Forward transliteration — or simply transliteration — is transliterating a word from one



(a) phonetic−based transliteration



(b) spelling−based transliteration

Figure 2.5: A graphical representation of phonetic-based and spelling-based transliteration approaches, where I represents a phonetical representation of the source (S) and target (T) words.

language to a foreign language. For example, forward transliteration of a Persian name "پروین" /pæɾviːn/ to English is "Parvin". Backward transliteration or *back-transliteration* is transliterating an out-of-dictionary word from its transliterated version back to the language of origin. For example, back-transliteration of "Parvin" from English to Persian is "پروین". Forward transliteration allows for creativity of the transliterator whereby multiple variants can be equally valid; back-transliteration however, is strict and expects the same initial word to be generated.

Automatic transliteration has been studied between English and several other languages, including Arabic [Stalls and Knight, 1998; AbdulJaleel and Larkey, 2003; Sherif and Kondrak, 2007a; Freitag and Khadivi, 2007; Kashani et al., 2007], Korean [Jung et al., 2000; Kang and Kim, 2000; Oh and Choi, 2002; 2005], Chinese [Wan and Verspoor, 1998; Xu et al., 2006], Japanese [Knight and Graehl, 1998; Goto et al., 2004; Bilac and Tanaka, 2005; Oh and Choi, 2006; Aramaki et al., 2007; 2008], and the romantic languages [Lindén, 2005; Toivonen et al., 2005]. Transliteration approaches based on the script of languages can be classified to those methods proposed for languages with Latin script, languages with symbolic script, or languages with Arabic script. Most research for languages with similar scripts is devoted to cross-lingual spelling variants, and their application in search tasks. Transliteration between

languages that are widely different in script is generally more challenging.

Another categorization of transliteration approaches is based on the information sources used in the process, which clearly distinguishes the methods proposed in the literature: approaches that consider the task a purely phonetical process and therefore use phonetics; approaches which perceive it an orthographic process and use spelling; and approaches that mix these two groups for a hybrid or combined approach.

We use the three categories of phonetic-based, spelling-based, and hybrid to review the literature on generative machine transliteration.

2.3.1 Phonetic-based Methods

Most early studies on transliteration applied speech recognition methods, and studied transliteration in a phoneme-based framework. The intuition behind this category of approaches is that phonetical representation is common among all the languages, which makes it possible to use it as an intermediate form between source and target languages (similar to interlingua MT). The other reason for interest in phonetic-based approaches (also known as *pivot* or phoneme-based) is the nature of the task; transliteration is a phonetical translation and phonetics can capture the pronunciation of the words. A general diagram of a phonetic-based approach is shown in Figure 2.5 (a). Phonetic-based methods identify phonemes in the source word S, and then map the phonetical representation of those phonemes (I) to character representations in the target language to generate the target word T. Different methods differ in their approaches of forming transformation rules ($\hat{S} \rightarrow \hat{I}$ and $\hat{I} \rightarrow \hat{T}$ based on Definition 3), and how phonemes or phonetical units of the source and target words are detected. We review these approaches in order of appearance, illustrating the main components of their generative transliteration system.

In general, phonetic-based systems borrow their transformation concepts from speech recognition phoneme-to-grapheme and grapheme-to-phoneme rule generation. Examples of such transformation rules for English spelling to phonetics are [Divay and Vitale, 1997]:

c → [k] / – {a,o},
c → [s].

This set of rules are read as: the grapheme “c” sounds as [k] if it is followed by “a” or “o”,

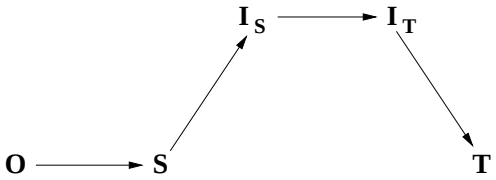
and it sounds [s] otherwise. Detecting phonemes of the words being processed is also part of these systems and directly affects the accuracy of these rules.

Arbabi et al. [1994] developed an Arabic-English transliteration system using knowledge-based systems and neural networks for pre-processing of the source Arabic words. Arabic names were input from an OCR (optical character recognition) system, which in turn was fed with phone-book entries. A knowledge-based system vowelised these names to add the missing short vowels, and passed them to a neural network to determine whether they are reliable in terms of Arabic syllabification. If reliable, then these names were converted to their phonetical representation using fixed transformation rules stored in a table. The phonetical representation was then transformed to English script using another set of fixed rules. Comparing this system with the outline in Figure 2.4, there is no formal transliteration training component, and only one transliteration component exists that performs vowelisation as a pre-processing of the source word S. The transliteration model was therefore pre-defined in the form of fixed transformation rules. The main drawback of this study is that the importance of forming transformation rules is ignored. The emphasis was vowelisation of the names and separating Arabic and non-Arabic names through syllabification process.

In contrast to the perception of Arbabi et al. [1994] who under-estimated transformation rule generation, this task is non-trivial. Divay and Vitale [1997] investigated generation of phoneme-to-grapheme transformation rules (also known as letter-to-sound rules), its challenges, and applications. Pronunciation of words in any language is determined by many parameters. For example, the position of words (morphophonemics) can determine how they are pronounced; also elision or epenthesis can make the pronunciation different from the orthographic presentation. Since the origin of the languages that proper names come from can vary, the correspondence of the written names and their pronunciation can be very hard to specify, and in some cases they differ substantially from the spelling [Divay and Vitale, 1997]. In some studies the problem of determining the diversity of proper names in terms of ethnic groups they belong to was studied by classifying them to their language group, or language family. This process increased the accuracy of systems that generate the grapheme-to-phoneme rules from proper names.

Knight and Graehl [1997; 1998] studied back-transliteration of Japanese out-of-dictionary words to English. The approach is phoneme-based with four main steps to be taken. Fig-

ure 2.5 (a) showed a general transliteration system that bridges between the languages using phonetical representation. In their proposed system, Knight and Graehl [1998] defined four main steps to be followed as shown below.



Japanese source word O was first transformed into electronic representation S using OCR. S is then transformed to its phonetical presentation I_S, then source phonemes mapped to target English phonemes I_T, and a phoneme-to-grapheme mapping generates the target English word T. Therefore, their model was formulated as:

$$T = \text{argmax}_T P(T) \sum_{S,I_S,I_T} P(O|S).P(S|I_S).P(I_S|I_T).P(I_T|T), \tag{2.11}$$

where $P(O|S)$ introduces the misspellings caused by OCR (input from katakana), $P(S|I_S)$ is the probability of pronouncing the source word, $P(I_S|I_T)$ converts the source sounds to the target sounds, $P(I_T|T)$ is probability of generating the written T from the pronunciation in I_T, and $P(T)$ is probability of having a sequence T in the target language.

To perform the calculations, Knight and Graehl [1998] used a weighted finite-state transducer (WFST) and weighted finite-state acceptor (WFSA). A finite state machine (FSM) is a model of behaviour composed of a finite number of states, transitions between those states, and actions defined by performing the transitions. A weighted finite-state transducer is a kind of FSM which defines three parameters for each transition: input, output, and weight. A weighted finite-state acceptor has only one input symbol and a weight for each transition between the states, and specifies which outputs the sequences that are more probable than others. To match such a model with the transformation rules of a transliteration model (Definition 3), each transition can be considered as a transformation rule with the source and target mapping to input and output, and probability mapping to a weight. In their model, Knight and Graehl [1998] implemented $P(T)$ using WFSA and the rest of the probabilities given in Equation 2.11 using WFST. To generate the best transliterations using WFSA, they implemented Dijkstra's shortest path and k-shortest paths algorithms [Eppstein, 1998] (TOP-K transliterations). The target language model implemented in $P(T)$ was

a unigram model made from the Wall Street Journal corpus, an on-line English name list, and an on-line gazetteer of place names. An English sounds inventory was taken from the CMU pronunciation dictionary. $P(I_S|I_T)$ was calculated based on the frequency information taken from the alignment of 8,000 pairs of English and Japanese sound sequences learnt using the estimation-maximisation (EM) algorithm. In comparison to the base system in Figure 2.4 on page 32, their system included both components, with WFSA and WFSTs built automatically and manually in the training stage, and then transferred as a transliteration model to the transliteration stage.

The English-Japanese model proposed in this study was strictly one-to-many. Such a model accommodates vowels that are often generated in a Japanese word after each occurrence of an English consonant (to avoid consonant clusters). In this model, mapping a sequence of characters in Japanese to only one English character is possible; this means that the model does not work in the reverse direction.

Knight and Graehl [1998] evaluated their automatic back-transliterator in two sets of experiments. One used 222 katakana phrases; however, no evaluation result was reported for this experiment because they considered the task difficult to judge: some of the input phrases were onomatopoetic (words or terms that imitate the sound it is describing) and some were even difficult for humans to transliterate. The other experiment was on 100 names of U.S. politicians taken from katakana. They compared their system's performance with four human transliterators — English native speakers — performing the same task. Human transliterators in general performed very poorly in comparison to the machine (24% versus 64% word accuracy). The reason for the low accuracy of humans, however, was their lack of information of Japanese phonetics.

Stalls and Knight [1998] proposed a similar method for back-transliteration of Arabic out-of-dictionary words into English. The challenges for Arabic language is much greater than for Japanese, as no specific pronunciation dictionary that covers all the out-of-dictionary words from different origins (not just English), short vowels are not written in Arabic, and there is a lack of resources for Arabic pronunciation. The transliteration system was evaluated on a test corpus of 2,800 names that resulted in 32.1% TOP-1 accuracy. The reason for failure of back-transliterating some of the names was their non-English origin, which was not reflected in the pronunciation conversion models.

Wan and Verspoor [1998] investigated a method of English-Chinese transliteration using the general approach of transforming the English name to its phonetical representation, and transforming the phonetics to Chinese writing. Since the phoneme-to-grapheme process is considered the most problematic and least accurate step, they limited their model to place names only, to reduce the variety. Since some place-names were partially translated, a pre-processing step (Figure 2.4) that performed dictionary look-up was used to detect those parts. A syllabification step segmented the English words to syllables based on the consonant boundaries. A sub-syllabification step further divided the syllables into sub-syllables to make them pronounceable within the Chinese phonemic set. Using a fixed English phoneme to Chinese, the phonetic representation of each sub-syllable is mapped to Chinese Pinyin, which is the most common standard Mandarin Romanization system. Pinyin uses the Latin script to represent sounds in standard Mandarin. Another fixed set of rules transforms Pinyin to Han (Chinese script). Therefore, the transliteration models were divided into a grapheme-to-phoneme step, and a phoneme-to-grapheme transformation which was based on a fixed set of rules. There was no evaluation reported for this approach.

Jeong et al. [1999] reported a method of back-transliteration for Korean out-of-dictionary phrases to English. Their study was divided into two main parts: identification of foreign words from Korean texts, and back-transliteration of them to English. The first step was extraction of non-Korean words using the statistics of the phonetical differences between Korean words and transliterated words. In the second step, they generated the back-transliteration candidates using a HMM model implemented as a feed-forward network with error-propagation. The transformation hierarchy was similar to Figure 2.5 (a). That is, only one level of phonetical presentation was considered. The main formula of ranking the candidates was

$$
\begin{aligned}
T &= argmax_T P(T|S) \\
&= argmax_T P(t_1 t_2 \dots t_m | s_1 s_2 \dots s_l) \\
&= argmax_T P(t_1 t_2 \dots t_m) \times P(s_1 s_2 \dots s_l | t_1 t_2 \dots t_m) \\
&= argmax_T P(I_1 I_2 \dots I_m) \times P(s_1 s_2 \dots s_l | I_1 I_2 \dots I_m) \\
&\cong argmax_T \prod_j P(I_{t_j} | I_{t_{j-1}}) \times P(s_j | t_j),
\end{aligned}
$$

where $\prod_j P(I_{t_j}|I_{t_{j-1}})$ shows the transition probability between two consecutive states in the HMM. In their model, Jeong et al. [1999] assumed that any Korean letter is only dependent on one single pronunciation unit in English. Their HMM model also considered only one-to-one relationships of characters. At the final stage, the candidate transliterations were compared against an English dictionary using similarity measures to prune the list of suggestions and rank them. They evaluated their transliteration accuracy in isolation and in an information retrieval framework. A bilingual corpus of 1,200 pairs was used by dividing that into training set of 1,100 pairs, and 100 for testing (no cross-validation). They reported TOP-1 accuracy of 47% and TOP-10 was 93%. The method resulted in 56% TOP-1 and 76% TOP-10 when dictionary matching was applied.

Jung et al. [2000] also proposed a method of English-Korean transliteration using an extended Markov window. They used the steps shown in Figure 2.5 (a) where English word pronunciations were taken from the Oxford dictionary, then a predefined set of rules mapped the syllabified phonetic units to Korean. A heuristic method of syllabification and alignment was proposed to assign probabilities to the set of mapping rules (training stage). In the transliteration stage, they generated all the possible syllables of each English word based on the consonant and vowel boundaries in the equivalent phonetical shape (pre-processing and segmentation steps), then transliteration generation started. The transliteration model was based on an extended Markov window. Based on a general formula:

$$T = argmax_T P(S)P(T|S),$$

they incorporated the context in the target language into the probability calculations. Therefore, the model is extended as

$$T = argmax_T \prod_i \frac{P(t_i|s_{i-1}t_{i-1})P(s_i|t_i s_{i-1})P(s_{i+1}|t_i s_i)}{P(s_{i+1}|s_i)}.$$

Their method was evaluated on a corpus of 8,368 English-Korean word-pairs with each English word accompanied with one or more transliterations. Two measures of precision and recall were defined based on the word accuracy metric, and reported their results in TOP-10 candidates generated. They reported a word accuracy of 54.9% when training and testing words were separated.

Oh and Choi [2002] studied English-Korean transliteration using pronunciation and contextual rules. In their training stage, they aligned English pronunciation units taken from a pronunciation dictionary and aligned it to phonemes to find the probable correspondence between an English pronunciation unit and phoneme. Based on the pronunciation of the English word, a Korean word was generated. They used the word formation information in the form of prefix and postfix to separate English words of Greek origin. Their method is also referred to as correspondence-based transliteration [Oh and Choi, 2006].

Lin and Chen [2002] presented a method of back-transliteration for English and Chinese. Their study however cannot be classified as a generative transliteration. A modified Widrow-Hoff learning algorithm automatically captures the phonetic similarities from a bilingual transliteration corpus. Their automatic method of extracting the phonetic similarities outperforms pre-defined phonetic similarities modelled as fixed transformation rules. In their approach, Lin and Chen [2002] using pronunciation dictionary transformed both English and Chinese names to their IPA presentation (Appendix C) and then applied the similarly measure on the phoneme (a similarity scoring matrix).

Virga and Khudanpur [2003a;b] studied English-Chinese transliteration using phoneme presentation of English names. They used the Festival speech synthesis system to convert English name into phonemes and then extracted the sub-syllables to match to Chinese and pronunciations and then converted them into Chinese. The approach they proposed was similar to Wan and Verspoor [1998], with a difference that the correspondence between English and Chinese pronunciations were automatically captured using GIZA++. They evaluated their method in retrieval of Mandarin spoken documents from a topic detection and tracking (TDT) corpus using English text queries. There is no standard evaluation reported in their paper.

Gao et al. [2004a;b] investigated English-Chinese transliteration in a framework that did not follow the source-channel model, the most popular approach in the previous literature, and used a direct model. Comparing the two formulas of source channel

$$T = argmax_T P(S|T)P(T), \tag{2.12}$$

and direct

$$T = argmax_T P(T|S), \tag{2.13}$$

they argue that the former concentrates more on the well-formed target strings but it does not incorporates the neighbouring phonemes, and also it does not support many-to-one mapping between source and target. They therefore added the target language model to the direct transliteration formula as

$$T = argmax_T P(T|S)P(T), \tag{2.14}$$

to build their underlying model. They evaluated their model on 46,306 English-Chinese word-pairs extracted from LDC (Linguistic Data Consortium) named entity list using word accuracy and character accuracy metrics. Their results indicated that the direct model outperforms the source-channel model in their transliteration experiments.

In general, phoneme-based transliteration has a primary advantage of elevating the role of pronunciation in the transliteration process. However, multiple steps involved in the process – including all the transformations from grapheme-to-phoneme, phoneme-to-grapheme, and sometimes phoneme-to-phoneme – increases the chance of propagating errors.

2.3.2 Spelling-based Methods

While the main concern of phonetic approaches was finding the phonemes of the source word, substituting their phonetical representation, and then transferring them to written target language, spelling-based (also known as *direct* or grapheme-based) methods map groups of characters in the source word S directly to groups of characters in the target word T. A general diagram of a grapheme-based approach is shown in Figure 2.5 (b) on page 34. Immediately obvious is that the number of steps in the transliteration process is reduced from two, or in some approaches three, to one. Spelling-based approaches only rely on the statistical information that is obtainable from the characters of the words.

In this section, similar to phonetic-based approaches, we review the literature in order of appearance.

Kang and Kim [2000] proposed a similar method to Jeong et al. [1999] for English-Korean transliteration and back-transliteration using HMM. They approached the problem by the

source-channel general formula, whereby P(T) is approximated by the Markov first order dependence assumption:

$$P(T) = P(t_1) \prod_{i=2}^{m} P(t_i|t_{i-1}).$$

For each source word S all the possible phoneme sequences are built; that is, where there is no pronunciation available, any segmentation of the word is possible. Using all these segments, a network is created that can generate all the possible transliterations of the source word. To assign a probability to these possible transliterations, they applied substrings, extracted from the training data. The length of each substring (they called them phoneme chunk) was incorporated in the probabilities assigned to each transformation by multiplying them by length. Evaluation was carried out using word accuracy and character accuracy metrics on an English-Korean corpus of 1,650 word pairs, with a fixed 150 test set separated, a corpus of 7,185 word pairs, and a third corpus of 2,391 pairs. For English-Korean transliteration they obtained a maximum of 55.3% TOP-1 for the first corpus. Back-transliteration accuracy was 34.7%. Their second corpus resulted in a maximum of 58.3% word accuracy for forward transliteration, and 40.9% for back-transliteration.

Kang and Choi [2000] investigated English-Korean transliteration and back-transliteration using a new alignment algorithm and decision-tree learning. For English, 26 decision trees were learnt for each letter (26^2), and for Korean 46^2 decision trees were learnt for each letter. Transformation rules in decision trees considered a context of three past letters and three future letters of each character in a source word. They evaluated their system on a bilingual transliteration corpus of 7,000 pairs with 1,000 for testing and 6,000 for training. They reported word accuracy of 44.9% for transliteration using left and right context, and 34.2% for back-transliteration. When information gain — the default method of attribute selection for decision learning — was used, the results were 48.7% and 34.7%, respectively. In Chapter 3, we similarly examine the effect of applying both past and future context for our English-Persian and Persian-English transliteration task.

AbdulJaleel and Larkey [2003] studied English-Arabic transliteration using n-gram models. Their transliteration was performed based on the general formula in Equation 2.14 on page 42, with the target language model being a bigram model: $P(T) = \prod_i (t_i|t_{i-1})$. Their system follows all the stages shown in Figure 2.4 on page 32. Training aligns the word pairs from a bilingual transliteration corpus using GIZA++. Then, transformation rules are formed

and probabilities are assigned to them based on the frequencies in the corpus. They compared their system with a hand-crafted model that was constructed with carefully chosen rules as a baseline. Their system resulted in 69.3% TOP-1 word accuracy where the baseline hand-crafted system was 71.2% accurate, evaluated on a corpus of 815 word pairs taken from AFP Arabic corpus. The impact of transliteration was also evaluated in a cross-lingual information retrieval task. Since Arabic is the closest language — in terms of script — to Persian, in Chapter 3 we adapt this method for English-Persian and Persian-English transliteration as a baseline system.

Min et al. [2004] and Li et al. [2004] in two studies which applied a similar approach, proposed a direct transliteration and back-transliteration method for English-Chinese and English-Japanese language pairs. They investigated an orthographic alignment process to derive the aligned transliteration units from a bilingual dictionary. In their approach, alignment was introduced using the source-channel model as

$$P(S,T) = \prod_{k} P(< \hat{S}, \hat{T} >_k \mid < \hat{S}, \hat{T} >_{k-n+1}^{k-1}) \tag{2.15}$$

where $< \hat{S}, \hat{T} >$ represents alignment between two substrings of the source and target words. Each alignment k of $< \hat{S}, \hat{T} >_k$ in a sequence of alignments is approximated using its last n alignments $< \hat{S}, \hat{T} >_{k-n+1}^{k-1}$. On a corpus of 28,632 unique word pairs, they reported word accuracy of 46.9% TOP-1 for English-Chinese transliteration when only unigrams were used. Increasing the context had a negative impact on their results.

Lindén [2005] investigated the problem of transliteration between romantic languages particularly for cross-lingual information retrieval. The approach used the direct model in Equation 2.13 on page 42 which considered a past and future context in the source word to predict a target word character in an n-gram based framework:

$$P(T|S) = \prod_{i} P(t_i | s_{i-2} s_{i-1} s_i s_{i+1}).$$

This model was implemented using a WFST and tested on 1,617 words in Finnish, Danish, Dutch, English, French, German, Italian, Portuguese, and Spanish. The system was evaluated using specially defined precision and recall metrics, which makes the comparisons with other studies difficult. The Finnish data, however, was used only to check the robustness

of the system and only added after the system was trained on other languages. The proposed model was particularly designed to extract cross lingual spelling variants, and was therefore tested for such a task as well. With the proposed approach being reliable for different languages, all from the Indo-European family of languages, we adapted a similar approach for our transliteration tasks in Chapter 3 as baseline.

Ekbal et al. [2006] investigated a revised joint source-channel proposed by Min et al. [2004] and Li et al. [2004] for Bengali-English. Transliteration units in the source word were chosen using a regular expression based on consonants, vowels, and matra (a Bengali language writing delimiter). They examined differing past and future context in their model, taken from the Equation 2.15, and context in the target word. To count for one-to-many alignments between English and Bengali, they provided hand-crafted transformation rules to their system. In case of failure in alignment even when incorporating handcrafted rules, manual intervention in the training phase fixed the errors. Once the training was complete, the transliteration model was ready for transliteration phase. They evaluated their system using a corpus of 6,000 people's names with 1,200 for testing and 4,755 for training (open test set). Their best model achieved 69.3% TOP-1 word accuracy for Bengali-English and 67.9% for the back-transliteration task.

Malik [2006] proposed a system of converting a word between two scripts of Punjabi: Shahmukhi, which is based on Arabic script, to Gurmukhi, which is a derivation of Landa, Shardha and Takri. The transliteration system used hand-crafted transliteration rules in two categories of character mappings and dependency rules. Dependency rules were contextual rules that define special cases of failure in simple character mappings. For evaluation, 45,420 words from classical and modern literature were extracted with an average transliteration accuracy of 98.95%.

Sherif and Kondrak [2007a] investigated Arabic-English transliteration using dynamic programming and substring-based transducer approaches. To count for many-to-many mapping of source and target words that occurs in transliteration (and had been ignored in the past studies), they applied phrase-based approaches of MT. Two methods were examined: monotone search using a Viterbi substring decoder, and a substring transducer. The advantage of the substring transducer is mentioned to be its capability in implementing a word unigram language model, it eliminates low probability mappings, and it handles NULLs

implicitly and therefore reduces the confusion NULLs may cause on the transducer. They evaluated their system on a training corpus of 2,844 word pairs, testing 300 word pairs; the language model was trained separately on 10,991 (4,494 unique) word pairs. Their results using word accuracy are reported only for seen data; that is, some of the training and testing data overlapped. Other studies also have used this evaluation paradigm; however, since the aim of a generative transliteration system is to transliterate unseen, newly appearing names, this method of evaluation seems unsatisfactory.

Li et al. [2007] proposed a transliteration method for personal names called semantic transliteration. By semantic, they meant language of origin, gender, and given or surname information of the source names. Their transliteration model was therefore formed as

$$P(T|S) = \sum P(T|S, l, g)P(l, g|S),$$

where l represents the language of origin and g represents gender. If any of the information in the model was missing then they removed that information source from their model. In their experiments three corpora were used with three languages of origin: Japanese, Chinese, and English. Names were separated to surname, female given name, and male given name. Using sequences of four characters, the origin of these names and their gender were detected. Corpora used were reported with 30,000 pairs for Japanese-Chinese, 34,600 for Chinese-Chinese, and 20,600 for English-Chinese. The performance of their system was reported using mean reciprocal rank, word accuracy, and character accuracy. The best overall accuracies achieved were 49.4% word accuracy, and 69.2% character accuracy. Although improvements were gained in comparison to their baseline phonetic-based system, the accuracies were not as high as similar studies which did not consider semantic information for English-Chinese transliteration.

2.3.3 Hybrid Methods

The phonetic-based and spelling-based transliteration approaches reviewed in the last two sections were investigated in two separate categories. Researchers have also considered a combination of these two categories as a third option. Phonetic-based approaches, having extra steps, are in general reported to be more error-prone than their spelling-based counterparts, and typically success rates of purely phonetic-based approaches are lower than spelling-based

methods, particularly for Arabic script languages. However, although spelling-based methods are more successful than phonetic-based approaches, they do not handle transliterations that have pronunciation widely different from the spelling. For example, the English place name "Edinburgh" is pronounced /'ɛdnbɜrə/ with "gh" sounds different from its normal pronunciation. In this section, an overview of hybrid approaches is reported; these aim to incorporate the strength of each category for higher overall accuracy.

Al-Onaizan and Knight [2002a] and [2002b] studied Arabic-English transliteration using both phonetic and spelling information. The hybridisation is based on a linear combination of the probabilities of these two methods:

$$P(T|S) = \lambda P_s(T|S) + (1 - \lambda)P_p(T|S),$$

where $P_s(T|S)$ represents the probability given by spelling approach and $P_p(T|S)$ is the score from phonetic approach. The spelling approach followed the source-channel model using Equation 2.12. The phonetic probability was adapted from Stalls and Knight [1998]. This approach was only proposed and evaluated for names of people, however. Names of locations and organisations which can be partly translated and partly transliterated were handled differently. Their evaluations showed improvement (11.9% in comparison to phonetic-based method, but a decline of 3.7% in accuracy in comparison to spelling-based method) in word accuracy using a hybrid method over phonetic-based methods in the first suggestion of the transliteration system (TOP-1).

Bilac and Tanaka [2004; 2005] demonstrated that back-transliteration accuracy can be improved by direct combination of spelling and pronunciation information. The difference of their work from the method proposed by Al-Onaizan and Knight [2002a;b] is that rather than producing back-transliterations based on spellings and phonetics independently, and then interpolating the results, they performed the combination during the transliteration process of each source word. They therefore proposed the following formula for a hybrid method:

$$P(\hat{T}_k|\hat{S}_k) = \alpha P_s(\hat{T}_k|\hat{S}_k) + \beta P_p(\hat{T}_k|\hat{S}_k),$$

where $\alpha + \beta = 1$.

Then, using the source channel formula in Equation 2.12 on page 2.12, they scored the transliterations for a ranked output. In their system, the alignment was performed using the

EM algorithm and following the Al-Onaizan and Knight [2002a] and [2002b] approach; the underlying transliteration model was kept as a WFST.

Evaluation of this system was performed on back-transliteration of Japanese and Chinese out-of-dictionary terms to English. They used a bilingual transliteration corpus taken from the EDICT dictionary including 714 word pairs with known pronunciation. The results showed 84.6% TOP-1 accuracy, for this corpus (without language model). Another corpus was taken from katakana comprising 150 word pairs with pronunciations extractable from the CMU dictionary resulted in 38.0% TOP-1 accuracy in comparison to 38.7% for the phonetic-based approach, and 32.7% for the spelling-based approach (without language model). Using a language model in their experiments resulted in no improvement, or small improvements. In general, evaluation on both Japanese and Chinese transliterations showed that direct combination for certain corpora had increased the accuracy.

Oh and Choi [2005], and Oh et al. [2006b] and [2006a] investigated a method of hybridisation of spelling- and phonetic-based approaches for English-Korean and English-Japanese transliteration. They criticised the hybrid models introduced so far for ignoring the dependence of the source word graphemes and phonemes whereas Oh and Choi [2002] considered this relation in their correspondence-based method. Other criticisms of the previous hybrid models were that they assigned a fixed weight to each of the spelling or phonetics approaches whereas, depending on the source word some are transliterated more phonetically and some are more based on the spelling. They therefore approached the transliteration problem by combining the spelling and phonetics, with consideration of correspondence information, in one model. Three machine learning algorithms were implemented to bring all these methods to one framework: a maximum entropy model, decision-tree learning, and memory-based learning. Transformation rules are learned using all the approaches (phonetics, spelling, correspondence, and a hybrid of phonetics and spelling) with a context length of three on each side of the transliteration unit that is mapped to the target substring.

Their evaluation results showed improvements in word accuracy in comparison to each of the other models independently. For English-Korean word accuracy was 68.4% for a corpus of 7,172 word pairs where 1,000 were chosen for testing. English-Japanese transliteration resulted in 62.3% word accuracy.

2.3.4 Combined Methods

System combination schemes have been shown to be successful for different natural language processing applications such as machine translation [Nomoto, 2004; Matusov et al., 2006; Rosti et al., 2007], part-of-speech tagging [Roth and Zelenko, 1998; van Halteren et al., 1998], parsers [Henderson and Brill, 1999; Nowson and Dale, 2007], word-sense disambiguation [Pedersen, 2000], and text categorisation [Larkey and Croft, 1996].

Combining multiple systems is usually performed in two framework of *glass-box* and *black-box*. *Glass-box* combination occurs when systems use details of their internal functionality in the combined system; hence combination happens before any final output is generated. An example of such a method for machine transliteration can be the linear combination of spelling- and phonetic-based methods as explained above, under hybrid methods literature. These approaches showed improvements in effectiveness of the transliteration systems in comparison to spelling-based or phonetic-based systems individually.

On the other hand, *black-box* combination works on the outputs of the systems, while the internal function of the systems is not altered [Huang and Papineni, 2007]. This method has been repeatedly applied in machine translation. Generally, combining systems is advantageous for two reasons: first, systems errors are independent and they do not propagate to each other [Bangalore et al., 2001]; and, second, each of the systems has its own efficacy, and combining accumulates these to the final system. However, weak methods may dilute the performance of the final system, so system selection is crucial.

Combined approaches have only recently been introduced to machine transliteration, leaving room for further studies. Oh and Isahara [2007a] studied English-Korean and English-Japanese transliteration using a combination of transliteration systems. They proposed a method based on support vector machines (SVM) and maximum entropy models (MEM) to re-rank the outputs of individual transliteration systems. These individual systems were from a variety of spelling-based, phonetic-based, and hybrid methods. Both their machine learning components, SVM and MEM, were trained on confidence score, language model, and Web frequency features. A confidence score was the probability (pr_j) assigned to each generated target word T_j in the list of (T_j, pr_j) transliterations that each system produced. However, it is not clearly explained how these scores are comparable across different systems. The Web frequency parameter was adapted from other Web-based systems, similar to the method

proposed by Al-Onaizan and Knight [2002b] which counts the co-occurrence frequencies of the transliteration pair on the Web.

For evaluation of their combined method, Oh and Isahara [2007a] used two corpora: An English-Korean corpus which consisted of 7,172 pairs, and an English-Japanese corpus which consisted of 10,417 pairs from the EDICT dictionary. Both corpora contained proper names, technical terms, and general terms. In their experiments, a fixed set of training and testing sub-corpora were used for evaluations (no cross-validation). Using seven individual systems, they reported 87.4% TOP-1 word accuracy for English-Japanese transliteration, and 87.5% for English-Korean, when the MEM-based approach is used. For the SVM-based approach these results were 87.8% and 88.2%, respectively.

To summarise the literature in generative machine transliteration, we observe a move towards phonetic-based approaches in early 90s when the first papers on automatic transliteration were published. These approaches evolved through the years, but their demand for pronunciation resources and language-dependant grapheme-to-phoneme or phoneme-to-grapheme conversion systems made them less appealing. Spelling-based approaches, on the other hand, have been more successful, and combining the two has mixed results. In this thesis, we study grapheme-based approaches for English and Persian. Persian uses a modified Arabic script which means short vowels are often omitted in writing. This feature also makes it hard to adapt a phonetic-based approach for Persian.

A general overview of the methods, corpora used, and accuracies reported in the reviewed literature is shown in Tables 2.1, 2.2, and 2.3 for manual (systems which used handcrafted transformation rules) and phonetic-based, spelling-based, and hybrid systems, respectively. Two main problems affecting most these studies that can clearly be seen in these tables are: first, the corpus specifications are overlooked in most these studies, with the majority reporting only size of each corpus; and second, while word accuracy was the most-reported measure, some studies used other measures, making comparisons across studies difficult.

2.4 Approaches of Transliteration Extraction

Learning *translation equivalents* from parallel or comparable corpus (Definitions 1 and 2) has been studied for machine translation for more than a decade. Statistical machine translation,

Method	Corpus Specification	Performance (%)
Handcrafted Rules		
Arabic-English [Arbabi et al., 1994]	phone-book entries, size unknown	unreported
Shahmukhi-Gurmukhi [Malik, 2006]	words from literature, 45,420	99, word accuracy
Phonetic-based		
Japanese-English (b) [Knight and Graehl, 1998]	people names, 100	64, word accuracy
Arabic-English (b) [Stalls and Knight, 1998]	names (type unknown), 900	32, word accuracy
English-Chinese [Wan and Verspoor, 1998]	unreported	unreported
Korean-English (b) [Jeong et al., 1999]	type or source unknown, 1,200	56, word accuracy
English-Korean [Jung et al., 2000]	type or source unknown, 8,368	55, word accuracy
English-Korean [Oh and Choi, 2002]	type or source unknown, 7,185	52, word accuracy 92, character accuracy
Chinese-English (b) [Lin and Chen, 2002]	names (type unknown), 1,574	83, mean reciprocal rank
English-Chinese [Virga and Khudanpur, 2003b]	unreported	unreported
English-Chinese [Gao et al., 2004a]	LDC corpus, 46,306	36, word accuracy 77, character accuracy

Table 2.1: An overview of accuracy of different transliteration methods in the literature (manual and phonetic-based). Note the accuracies reported in this table are the best reported amongst all the experiments in the corresponding papers. (b) indicates back-transliteration.

Method	Corpus Specification	Performance (%)
English-Korean (f,b) [Kang and Kim, 2000]	type or source unknown, 7,185	58, word accuracy (f) 41, word accuracy (b)
English-Korean (f,b) [Kang and Choi, 2000]	type or source unknown, 7,000	48, word accuracy (f) 35, word accuracy (b)
English-Arabic [AbdulJaleel and Larkey, 2003]	extracted from AFP corpus, 815	69, word accuracy
English-Chinese, [Min et al., 2004]	type or source unknown, 28,632	47, word accuracy
9 European languages [Lindén, 2005]	dictionary, 1,617	70, precision defined using reciprocal rank
Bengali-English [Ekbal et al., 2006]	people names, 6,000	68, word accuracy
Arabic-English [Sherif and Kondrak, 2007a]	type or source unknown, 3,144	2.01, avg. edit distance
English-Chinese [Li et al., 2007]	different language origins, people names were gender seperated	58, mean reciprocal rank 47, word accuracy 67, character accuracy

Table 2.2: An overview of accuracy of different transliteration methods in the literature (spelling-based). Note the accuracies reported in this table are the best reported amongst all the experiments in the corresponding papers. (b) indicates back-transliteration, and (f) represents forward transliteration.

Method	Corpus Specification	Performance (%)
Hybrid (glass-box)		
Arabic-English	names of locations and organisations,	73, word accuracy
[Al-Onaizan and Knight, 2002b]	unknown size	
Japanese-English (b)	EDICT dictionary,714	85, word accuracy
Chinese-English (b)	katakana words, 150	38, word accuracy
[Bilac and Tanaka, 2004; 2005]		
English-Korean	7,172	68, word accuracy
English-Japanese	EDICT,10,417	62, word accuracy
[Oh et al., 2006a]		
Combined (black-box)		
English-Korean	7,172	87, word accuracy
English-Japanese	EDICT,10,417	88, word accuracy
Oh and Isahara [2007a]	names, technical terms, and general terms	

Table 2.3: An overview of accuracy of different transliteration methods in the literature (hybrid and combined). Note the accuracies reported in this table are the best reported amongst all the experiments in the corresponding papers. (b) indicates back-transliteration, and (f) represents forward transliteration.

in particular, is reliant on this process to learn translation by examples [Brown et al., 1990; Melamed, 2000]. Proper names and technical terms in these texts need special attention; most of them rarely appear in documents and therefore are often undiscovered by methods that rely only on co-occurrence frequencies. Transliteration extraction, therefore, emerged as a study on methods of extracting transliteration terms, and consequently enriching translation lexicons.

Transliteration extraction studies in the 90s — formerly known and reported as named entity translation — were heavily influenced by machine translation techniques, especially statistical word alignment methods. Researchers, following the tradition of the MT community, started using parallel corpus. Later, the lack of parallel corpus — a rare resource for many languages — led to the exploration of approaches that benefit from comparable corpus,

bilingual dictionaries, and nowadays, the Web. The main metrics of evaluation in the field of translation and transliteration extraction is precision (the percentage of correct correspondences found among words or phrases) and recall (the percentage of found correspondences between words or phrases). Details of some of these studies are reviewed in this section.

Brown et al. [1990] first introduced sentence alignment on parallel corpus; all the other studies in machine translation [Brown et al., 1993], and bilingual lexicography [Catizone et al., 1989] on parallel texts were subsequent. Gale and Church [1991] introduced word correspondence in parallel text. They argued that in aligned sentences, word order is not preserved and, therefore, the term *alignment* should only be used at the sentence level, and *correspondence* should be used at the word level. Replacing the probabilistic transfer dictionary (built using the EM algorithm) that was used by Brown et al. [1990] with a contingency table, Gale and Church [1991] proposed using similarity measures (in particular ϕ^2, a χ^2-like measure) to find association of words in aligned sentences. They applied their method on English-French data and used the morphological resemblance of these two languages to increase the precision of word correspondence.

Van der Eijk [1993] proposed the acquisition of bilingual lists of terminological expressions from a Dutch-English parallel corpus. The main strength of his work was considering phrases instead of words. Part-of-speech tagging, co-occurrence statistics, and the position of the terms were used in this study.

Following the successful word and phrase alignment studies, translation of technical terms became a popular topic. Unfamiliarity of translators with domain-specific terms which cannot be found in most dictionaries motivated researchers to automatically extract those terms and their equivalents in other languages, and augment them to dictionaries. Dagan and Church [1994], in an early attempt at extracting transliteration equivalents from parallel corpus, developed a tool called *Termight* that semi-automatically extracts technical terms and their translations. This tool relied on part-of-speech (POS) tagging and word-alignment to extract candidate pairs, and the user was responsible for filtering. In this study increasing recall, and therefore extracting less-frequent equivalents that word-aligners would miss was the main goal.

In contrast to the previous studies on word alignment in parallel corpus, Rapp [1995] considered the correlation between the co-occurrences of words in non-parallel, comparable,

English-German news documents. He showed that even in comparable texts, patterns of word co-occurrence strongly correlate. This study was a basis for further consideration of comparable corpus — instead of hard-to-find parallel resources — in the field, both for machine translation and transliteration.

Later, Fung and McKeown [1997] continued the trend of technical-term translation on noisy parallel documents (English-Japanese and English-Chinese): documents that had no potential of getting aligned at the sentence-level. Technical terms were extracted using the Justeson and Katz [1995] technical term finder based on the order of POS tags (technical terms are either adjective-noun or noun-noun multi-words). They proposed using dynamic recency vectors instead of absolute word positions in texts to find the similarity of the terms. Using these vectors, they formed spectral signals for all the words in texts, and then used pattern matching on those signals recognised translations of the words. In their work, proper names and low-frequency terms were excluded. Although their work was focused on technical terms, no transliteration feature was considered; thus it can be categorised to the machine translation area.

Nagata et al. [2001] proposed a method of technical term translation extraction for English and Japanese. Using partial translations of terms in Web documents — documents that contain translations of some phrases immediately after their first occurrence — they extract a list of English-Japanese technical terms. The most important clue for distinguishing these partial translations were original words that occur in parentheses in front of their translations. Use of table aligned terms and term co-occurrence probabilities were also examined. Their experiments also showed that mining the Web for technical term translations is most effective for the fields of computer science, aeronautics, and law.

One of the first studies to target proper names in particular is by Fung and Yee [1998]. They proposed a transliteration extraction method on comparable corpus. Handling English-Chinese news documents, they used information retrieval techniques such as the vector space model [Salton, 1989] — and similarity measures based on term frequency (*tf*), and inverse document frequency (*idf*) — to find transliteration instances. Similar to Fung and McKeown [1997], they used a bilingual lexicon as seed words to start with, and then augmented it with the newly found pairs. Their algorithm was high is precision, but low in recall. Precision was defined as the number of correct transliterations found, and recall was the number of

transliteration pairs existed in the bilingual corpora.

Huang and Vogel [2002] investigated named entity translation of English and Chinese, with an emphasis on proper names (persons, locations, and organisations). In their proposed approach, they used a commercial named entity annotator system to extract named entities and their type from a parallel sentence aligned corpus. The candidate translations were chosen using co-occurrence frequencies and translation probabilities calculated based on the Brown et al. [1993] models. The technique was an iterative approach that started from initial annotations, refined annotations, and gradually created a dictionary of transliterations from the corpora. The main contribution of this work was its focus on named entities; however, it required a parallel corpus that is not easy to obtain for many languages.

A completely different approach of transliteration discovery was proposed by Al-Onaizan and Knight [2002b]. Their method required neither parallel, nor comparable corpora of bilingual documents. Although they still entitled their work with named entity translation, the transliteration nature of the task was considered and applied in the process. In order to build an Arabic-English named-entity dictionary, they first specified the named entity phrases and separated person names from location and organisation names. A transliteration generation paradigm[3] (that used both phonetic and spelling features), generated a ranked list of suggested transliterations. Candidates were then re-scored using straight Web counts, co-references, and contextual Web counts. They also used Web search for unsuccessful transliteration generations. The evaluation however was small scale and used only 20 Arabic newspaper articles for testing and 21 for training.

Similar to the Al-Onaizan and Knight [2002b] study, in more recent literature much of the attention is focused on the Web as a resource of discovering transliteration-pairs [Cheng et al., 2004; Masuyama and Nakagawa, 2005; Zhang et al., 2005; Chen and Chen, 2006]. Different clues were considered to find transliteration equivalents on the Web, for example Lu et al. [2002] used anchor text linked to target language equivalents [Lu et al., 2002], or Oh and Isahara [2007b] validated the output of their generative transliteration systems using the Web.

Other than Web-oriented solutions, traces of using phonetical and co-occurrence measures continue to be explored in recent literature. Lam et al. [2004], for example, used similarity

[3]Fully explained in Section 2.3

of English and Chinese proper names at the phoneme level. Again, in case of failure, they resorted to the Web to extract transliterations.

Sproat et al. [2006] reported their named entity transliteration on Chinese-English comparable corpus. In their method of extracting transliteration pairs, they used phonetical transliteration scores, a page-rank like scoring approach, co-occurrence frequencies, and temporal distribution of candidate pairs. Their dataset contained 234 Chinese documents and 322 English documents. In a similar approach, Klementiev and Roth [2006] used phonetic and temporal distribution to match English-Russian named entities. They proposed discriminative alignment of substrings of words to match the transliteration variants. Their experiments are reported on a corpus larger than its pioneers; it contained 2,327 Russian and 978 English news articles. Another example of recent studies that consider comparable corpus for transliteration extraction is the research of Alegria et al. [2006] on Basque-Spanish. Transliteration rules (or transformation rules) were manually made, and scores computed based on such rules. They also considered scores from Web counts.

Talvensaari et al. [2007] proposed a method for languages that share one script (Swedish and Finnish). They explored the use of skip-grams [Keskustalo et al., 2003] — or fuzzy matching — to align the words which were not found in a general-purpose dictionary. They managed to extract corresponding transliterations in comparable corpus which other frequency-based and time-based methods had failed to discover.

Lam et al. [2007] argue that transliteration is both semantic and phonetic. That is, there is always a possibility of transliterating one part of a name semantically, and another part phonetically. They therefore proposed a named entity matching model that makes use of both semantic and phonetic evidence. They also mined the Web for discovering new named entity translations from daily news, as an application of their matching algorithm. For the language pair of English-Chinese their approach shows effective discovery of unseen transliteration pairs.

Sherif and Kondrak [2007b] proposed a bootstrapping approach that uses a stochastic transducer for Arabic-English transliteration extraction. In their task of document-aligned named entity extraction, they used POS tagging for 1000 English documents, and refined their list of names extracted from English documents manually. Then, compared the performance of fuzzy matching and bootstrapping methods for a transliteration extraction task.

Their transducer essentially learns one-to-one relationships. The approach lacks context sensitivity.

In summary, transliteration extraction is rooted in studies on finding translation equivalents in MT. The importance of considering proper names were only discovered and highlighted in more recent studies. Discovery of transliteration equivalents started from the work on parallel corpus, then considered noisy-parallel texts. Comparable corpus appeared later, followed by a focus on the Web for a period of 3-4 years. Most recently, studies again tend to focus on comparable corpus in addition to the Web. Transliteration characteristics such as phonetical resemblance of transliteration equivalents were largely ignored in the past. Now, transliteration has found its place as a separate topic of study, focusing on spelling and phonetical properties of the transliterated pairs.

2.5 Methodology

This section provides methods and assumptions that underlie this study to provide an outline of the experiments performed and evaluations made in the following chapters.

2.5.1 English-Persian Transliteration Corpus

Generally, a bilingual transliteration corpus (Definition 5) is not readily available for transliteration studies, particularly for languages with relatively few computerised resources such as Persian. We therefore constructed an English-Persian corpus by taking 40,000 phrases from lists of English names found on the Web. These words were names of geographical places, people and companies. These phrases often consisted of more than one word which were split into words with duplicates discarded, resulting in a corpus of 16,670 word pairs. We refer to this corpus as B^{+}_{e2p}. This English corpus was then transliterated by 26 native Persian speakers. The transliterators, in terms of education level, were divided into three groups (bachelor's degree, masters, and PhD) as shown in Table 2.4. As can be seen in this table, all of the transliterators were university students or graduates.

Since the word selection process was not language-aware, and English script was the primary concern, many words of different language origins (such as Arabic, French, and Dutch) were included in the corpus. Depending on the transliterators' assumptions about the origin of a word, different characters may have been chosen for the same word. For

Level of Education (%)	Course of Study (%)
Bachelor (38.5)	Computer Science (44.4)
	Other (54.6)
	(Physics, Elec. Eng., Architecture,
	Nutrition, Lang., Psychology)
Masters (53.8)	Computer Science (100)
PhD (11.5)	Computer Science (100)

Table 2.4: Educational specifications of 26 transliterators of the B_{e2p} and B^{+}_{e2p} corpora.

example, if the transliterator assumed "John Pierre" was French, then for the character "j" the transliterator might have employed the Persian character ژ [ʒ], pronounced same as "su" in measure, instead of widely used ج [dʒ] which sounds as "j" in jet.

While all of these words chosen were in English script, they may have been transliterated from other scripts (for example, Arabic, and Chinese). We randomly chose a subset of 2,000 name pairs from the B^{+}_{e2p} collection, from which all names with origins from non-English script languages were removed. This resulted in a sub-corpus of 1,857 name pairs, which we call B_{e2p}.

2.5.2 Persian-English Transliteration Corpus

For Persian-English transliteration, we constructed the B_{p2e} corpus from a publicly available Web resource[4]. This corpus consists of 2,010 Persian person names with their widely-used English transliterations gathered from different sources. Each word is accompanied by one to five transliteration variants (provided by different people).

2.5.3 Multi-Transliterator Corpus

The corpus B^{+}_{e2p} was constructed using only one transliterator per English word, so does not allow rigorous study on the influence of language origin of source words and transliterators on the performance of transliteration systems. To determine the parameters affecting evaluation of transliteration systems, we required a carefully constructed corpus with following

[4] `http://cleo.lcs.psu.edu/`

controlled aspects: language of origin of source words, number of transliterators and language knowledge of transliterators. Therefore, we recruited seven transliterators (H_1, H_2, ... , H_7: all native Persian speakers from Iran) to transliterate 1500 proper names. They were instructed to provide just one transliteration per word. These source names were taken from lists of proper names all written in English on English Web sites. However, five hundred of these names had Arabic origin and five hundred were Dutch origin. The transliterators were not told of the origin of each word. The entire corpus, therefore, was easily separated into three sub-corpora of 500 words each based on the origin of each word. To distinguish these collections, we use E_7 to denote the English sub-corpus, D_7 for the Dutch sub-corpus, A_7 for the Arabic, and the whole 1500 word corpus as EDA_7. A portion of this corpus is shown in Appendix E.

Dutch and Arabic were chosen with an assumption that most Iranian Persian speakers have little knowledge of Dutch, while their familiarity with Arabic should be in the second rank after English. All of the participants held at least a Bachelors degree, with three having a Masters degree, and one a PhD. Table 2.5 summarizes the information about the transliterators and their perception of the given task. Participants were asked to scale the difficulty of the transliteration of each sub-corpus from easy to hard, and also to indicate their level of familiarity with foreign languages. Task difficulty was indicated as a scale from 1 (*hard*) to 3 (*easy*). Similarly, the participants' confidence in performing the task was rated from 1 (*no confidence*) to 3 (*quite confident*). The level of familiarity with second languages was also reported based on a scale of zero (*not familiar*) to 3 (*excellent knowledge*).

The information provided by participants confirms our assumption of transliterators' knowledge of second languages: good familiarity with English, some knowledge of Arabic, and no prior knowledge of Dutch. Also, the majority of them found the transliteration of English terms of medium difficulty, Dutch was considered mostly hard, and Arabic as easy to medium.

2.5.4 Evaluation Methodology

All the experiments in this thesis apply *10-fold cross-validation* in which the whole corpus is partitioned into ten disjoint segments. Then, for each run, one of the segments is used for testing and the rest for training. This method guarantees a fair test to avoid using the

	Second Language Knowledge				Difficulty,Confidence		
Transliterator	English	Dutch	Arabic	Other	English	Dutch	Arabic
H_1	2	0	1	-	1,1	1,2	2,3
H_2	2	0	2	-	2,2	2,3	3,3
H_3	2	0	1	-	2,2	1,2	2,2
H_4	2	0	1	-	2,2	2,1	3,3
H_5	2	0	2	Turkish	2,2	1,1	3,2
H_6	2	0	1	-	2,2	1,1	3,3
H_7	2	0	1	-	2,2	1,1	2,2

Table 2.5: Transliterator's language knowledge and perception of difficulty in creating the corpus. Language knowledge is scaled from 0 (not familiar) to 3 (fluent). Difficulty and confidence are scaled from 1 (hard) to 3 (easy).

easiest part of the corpus for testing [Allen, 1995].

To determine the significance of differences between the results of different methods, for all the experiments, a paired *t-test* is applied on the results of ten runs from cross-validation; a *p-value* is reported up to four significant digits, and if it is smaller than 0.0001 this is denoted as "< 0.0001".

Single-Transliterator Metrics

Results of our transliteration experiments are evaluated using the standard transliteration measures of word accuracy and character accuracy. The second measure is based on the edit distance between the system transliterated word and the expected transliteration. The edit distance measures the number of character insertions, deletions and substitutions that are required to transform one word into another [Levenshtein, 1965].

Word accuracy (A), also known as transliteration accuracy, measures the proportion of transliterations that are correct:

$$A = \frac{\text{number of correct transliterations}}{\text{total number of test words}}.$$

Word accuracy is reported for different cut-off values. For example, TOP-1 word accuracy indicates the proportion of words in the test set for which the correct transliteration was the first candidate answer returned by the transliteration system, while TOP-5 indicates the

proportion of words for which the correct transliteration was returned within the first five candidate answers.

In general, the appropriate cut-off value depends on the scenario in which the transliteration system is to be used. For example, in a machine translation application where only one target word can be inserted in the text to represent a source word, it is important that the word at the top of the system generated list of target words (by definition the most probable) is one of the words generated by a human in the test corpus. Alternately, in a cross-lingual information retrieval application, all variants of a source word might be required. For example, if a user searches for an English term "Tom" in Persian documents, the search engine should try and locate documents that contain both "تام" (3 letters:ت-ا-م) /tɒm/ and "تم" (2 letters: ت-م) /tɔm/, two possible transliterations of "Tom" that would be generated by human transliterators. In this case, a metric that counts the number of transliteration variants (T_k) that appear in the top n elements of the system generated list, L, might be appropriate.

Character accuracy, or character agreement, checks for the percentage of matched characters for each word pair:

$$CA = \frac{len(T) - ED(T, L(T_i))}{len(T)},$$

where, $len(T)$ is the length of the expected target word T, $L(T_i)$ is the suggested transliteration of the system in rank i, and ED is the edit distance between two strings [Hall and Dowling, 1980].

Multi-Transliterator Metrics

Specific experiments we designed for studying the parameters affecting the performance of transliteration systems and evaluation process demand for two types of metrics: metrics which evaluate systems, and metrics which measure subjects' influence on evaluation. We therefore introduce appropriate metrics below.

System Evaluation

As discussed earlier, in general, there are two commonly used metrics for transliteration evaluation that measure the accuracy of transliteration systems: word accuracy and character accuracy. However, when more than one transliteration is available for a given source word,

multiple variants need to be taken into account. Hence, we define three varieties of word accuracy: *uniform*, *majority* and *weighted*.

Uniform word accuracy (UWA) equally values all the transliteration variants provided for a source word. That is, if for a word-pair (S, T) that $T = \{T_i\}$ and $|T| > 1$, a transliteration system under evaluation generates any of T_i variants in T, it is counted in favour of that system.

Majority word accuracy (MWA) selects only one of the provided transliterations as valid. The criteria of choosing the preferred variant is that it must be suggested by the majority of human transliterators.

Weighted word accuracy (WWA) allocates a weight to each of the transliterations based on the number of times they are suggested by transliterators. In other words, all the transliteration variants are valid with a given weight.

Note the MWA and WWA differ from UWA only when duplicate transliterations are not removed from a testing corpus. In our experiments, these three measures are computed using the EDA_7 corpus for system evaluation in Chapter 6.

Human Evaluation

We define an agreement metric to evaluate the level of agreement between human transliterators based on *raw agreement* described by Mun and Eye (2004).

For any source word S_i, there are $|T_i|$ different transliterations made by the h_i human transliterators ($h_i = \sum_{j=1}^{|T_i|} h_{ij}$, where h_{ij} is the number of times source word S_i was transliterated into target word T_{ij}.) When any two transliterators agree on the same target word, there are two agreements being made: transliterator one agrees with transliterator two, and vice versa. In general, therefore, the total number of agreements made on source word S_i is

$$\sum_{j=1}^{|T_i|} h_{ij}(h_{ij} - 1).$$

Hence the total number of actual agreements made on the entire corpus of K words is

$$G_{act} = \sum_{i=1}^{K} \sum_{j=1}^{|T_i|} h_{ij}(h_{ij} - 1).$$

The total number of possible agreements – when all human transliterators agree on a single target word for each source word – is

$$G_{poss} = \sum_{i=1}^{K} h_i(h_i - 1).$$

Thus the proportion of overall agreement is defined as

$$P_G = \frac{G_{act}}{G_{poss}}. \tag{2.16}$$

2.6 Summary

This chapter covered literature and background information on two main topics in relation to this thesis: machine translation and machine transliteration. Literature on machine transliteration was divided into transliteration extraction from bilingual document corpus and transliteration generation. Transliteration generation was comprehensively reviewed to provide the background on the research problem studied in this thesis, highlighting their strengths and weaknesses. Outcomes of the research on generative transliteration suggested that spelling-based and combined approaches are stronger than phonetic-based methods. We build our transliteration techniques reported in Chapters 3 to 5 on this observation, developing spelling-based and combined methods for English and Persian. Also, studying these methods we identified their deficiencies in their evaluation. The excuse that creating a bilingual corpus is difficult has allowed publication of transliteration systems without any evaluation. Chapter 6 addresses this problem and provides solutions for a reliable evaluation process.

Chapter 3

Transliteration Based on N-grams

N-gram models, at both the character-level and word-level, have been extensively used in natural language text processing; transliteration is no exception. Most transliteration methods in the literature have borrowed the idea of fixed length character-level segmentation from n-grams, and also the concept of context knowledge. In this chapter, we present a performance comparison of existing spelling-based transliteration methods that employ n-grams for English-Persian and Persian-English transliteration. We also examine the role of past and future context, different backoff methods, and target language models.

3.1 Introduction

An n-gram is a sub-sequence of n elements from a given sequence. The elements in the sequence can be letters, words, or any basic item that is defined in the application. An n-gram model usually defines a context with a specified number of consecutive elements to consider, instead of using a single element at a time, and then, given the past $n-1$ elements predicts the n^{th} element in the sequence using statistical properties of n-grams. Another term generally used for these statistical models of sequences is *language model*. Conventionally, an n-gram of size one is called *unigram*, size two is a *bigram*, size three is a *trigram*, and size four or more is simply called an *n-gram*.

N-gram models have been used in various areas dealing with text data, such as NLP applications like part-of-speech tagging, natural language generation, and word similarity [Jurafsky and Martin, 2008], or other applications such as information retrieval, compression,

authorship identification, and optical character recognition. An advantage of such models is their flexibility to noisy data [Grossman and Frieder, 2004]. Character-level n-grams are beneficial to a character-based problem such as transliteration, and currently most of the spelling-based methods reported in the literature are substantiated by n-grams. The main idea is that, given $n-1$ context source characters, how to best transliterate the n^{th} character. Some studies, however, use context in target words as well as source words.

In this chapter, to address our main research problem around improving machine transliteration effectiveness for the language-pair English and Persian, we explore the idea of using common approaches of automatic transliteration for English-Persian and Persian-English transliteration. We therefore specifically focus on two research questions:

1. for the language pair English and Persian, which of the existing transliteration methods are more effective; and,

2. can we modify these existing approaches to improve the effectiveness of English-Persian and Persian-English transliterations?

We examine generic approaches based on n-grams proposed for transliteration of some other language-pairs — with some modifications — for our problem. The main difference between transliteration methods employing n-grams relies on their selection of context size. We therefore designed three main sets of experiments to study the effect of context on transliteration. By varying past and future context sizes we determine the best n-gram scheme for English-Persian and Persian-English transliterations. Different backoff approaches for those models that use both past and future contexts are investigated and compared. We also explore the effect of using a target language model. Our results indicate that using only one past context symbol is the most effective model for English-Persian transliteration, while for Persian-English transliteration using symmetric past and future context — especially two past and two future context symbols — gives the highest performance.

3.2 N-grams in the Transliteration Process

As explained in Chapter 2, the transliteration process consists of a training stage, running once on a bilingual training corpus $B = \{(S, T)\}$, and a transliteration generation stage

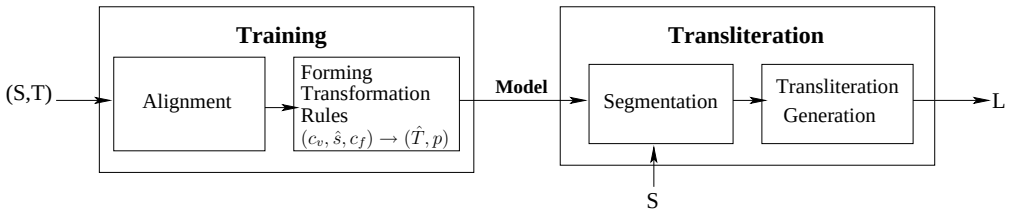


Figure 3.1: A general transliteration approach based on n-grams.

that produces a ranked list of target words $L = \{(T_i, pr_j)\}$ for each source word S. The sequence sketched in Figure 3.1 depicts the transliteration process. The training stage itself is composed of two tasks: alignment of source-target words, alignment-aided segmentation, and transformation rule generation using alignment provided clues. The transliteration stage also consists of two main tasks: segmentation of the test source word with the aid of transformation rules heads, and target word generation. In the following sub-sections, we demonstrate these tasks from the perspective of an n-gram based transliteration system.

3.2.1 Segmentation and Transformation Rule Generation

Whenever an input word is fed to a transliteration system, either for training or testing, it must be split into defined segments. A *segment* is a sequence of symbols of length n as specified in the n-gram model. For example, if we choose a bigram model, each segment contains two symbols. *Symbols* are individual characters, or groups of characters, in the source or target language alphabet; for example, the pair of English characters "ch" might be considered a single symbol. In a transliteration system, symbols can be derived using statistical character-alignment methods such as hidden Markov models [Jung et al., 2000], or translation models [Brown et al., 1993]. An alternative approach is to hand-craft the symbols of allowable source and target graphemes, as adopted by AbdulJaleel and Larkey [2003] for English to Arabic transliteration.

To clarify, consider the sample training set and English to Persian character-alignment of word-pairs in Figure 3.2. Note that Persian characters are written separately and from left-to-right for consistency with their English equivalent. Alignment, as explained in Chapter 2, relates transliterated characters in source and target words. From Figure 3.2 it can be seen

Training set:	**Alignment:**
(baird, بدری)	b a i r d | \ / | | ب ی ر د
(beers, بزری)	b e e r s | | | | | ب ی ε ر ز
(blasi, بلاسی)	b l a s i | | | | | ب ل ا س ی
(christopher, کریستفر)	c h r i s t o p h e r \ / | | | | | \ / | | ک ر ی س ت ε ف ε ر
(enrique, انریکو)	e n r i q u e | | | | | | | ا ن ر ی ک و ε
(schaffer, شافر)	s c h a f f e r \ | / | \ / | | ش ا ف ε ر
(smith,اسمیت)	s m i t h / \ | | \ / ا س م ی ت

Figure 3.2: Sample word-pairs and their alignments from an English-Persian transliteration process.

that multiple characters in English can share only one Persian character and vice versa. For example, in the source word "christopher", "p" and "h" align to "ف" [f] in Persian, and therefore they form a "ph" symbol. Similarly, "ai", "ch", "th", "sch" and "ff" are counted as symbols in the source language. On the other hand, "اس" /es/ (transliteration of "s" in "smith") is counted as a target language symbol.

Once the alignment process is completed and all symbols are identified, we proceed with segmentation. Each segment contains a context and a translatable symbol $\hat{s}$ (a substring transliterated to $\hat{T}$). Context size is defined based on the n-gram model used. As the general definition of transformation rule implies (Definition 3 on page 32), from a training stage of each transliteration system we expect to generate transformation rules in the form of $\hat{S} \rightarrow (\hat{T}, p)$. This rule is interpreted as segment $\hat{S}$ in source language being transliterated to $\hat{T}$ in target language with probability p. The concept of context is not limited to only past symbols. We generalise this to all symbols surrounding $\hat{s}$ and therefore, for brevity, we introduce the notation $v\backslash f$ to indicate that v previous source symbols and f future source symbols make up a context, that is:

$$v\backslash f = s_{i-v} \ldots s_{i-1} s_{i+1} \ldots s_{i+f}.$$

In other words, past symbols make a *v-gram* context and future symbols make a *f-gram* context. In order to avoid boundary conditions, we also assume that source word S is extended to the left and right as far as is required with a special symbol "#" that always transliterates to an empty symbol ε. Therefore, $[-v \ldots |S| + f]$ are all valid indexes into S. For the n-gram based transliteration systems only, we denote a transformation rule as $(c_v, \hat{s}, c_f) \rightarrow (\hat{T}, p)$ with $\hat{S} = c_v \hat{s} c_f$, where c_v is past context, c_f is future context, and $\hat{s}$ is a substring transliterated to $\hat{T}$ in context of c_v and c_f with probability of p (as shown in Figure 3.1).

The aligned source-target word pairs can be used to collect statistics on the frequency of occurrence of the chosen symbols. Therefore, probability p in the transformation rule is calculated by:

$$\begin{aligned} P(\hat{T}|\hat{S}) &= \frac{\text{freq}(\hat{T}|\hat{S})}{\text{freq}(\hat{S})} \\ &= \frac{\text{freq}(\hat{T}|c_v, \hat{s}, c_f)}{\text{freq}(c_v, \hat{s}, c_f)}, \end{aligned}$$

where, $\text{freq}(\hat{T}|c_v, \hat{s}, c_f)$ is the frequency with which $\hat{s}$ is aligned with $\hat{T}$, with $\hat{s}$ preceded by c_v and followed by c_f, and $\text{freq}(c_v, \hat{s}, c_f)$ is the frequency of $\hat{s}$ occurring in context $c_v\hat{s}c_f$. Since these frequencies are extracted from aligned pairs, they reflect the statistics of the transliterations registered in the training corpus B.

As an example for this process, consider a basic unigram based model which considers no context and is therefore denoted as 0\0. Based on the training set in Figure 3.2, 0\0 defines a set of valid source symbols or segments as

{b, a, i, r, d, ch, r, s, t, o, ph, e, q, u, sch, ff, m, th},

which form a set of the heads of transformation rules as $\{\hat{S}\}$ (later used in the transliteration stage for segmentation). Thus, from the first word-pair (beers, زری‌ب)[1], we derive these rules:

(–,b,–) → (ب , 1.0),
(–,e,–) → (ی , 0.17),
(–,e,–) → (ϵ , 0.67),
(–,r,–) → (ر , 1.0),
(–,s,–) → (ز , 0.25).

With the same training set, a bigram model that uses one past context symbol, 1\0, generates segments as

{#b, bai, air, rd, #ch, chr, rs, st, to, oph, phe, eq, qu, ue, #sch, scha, aff, ffe, er, #s, sm, mi, ith},

adding one padding character at the beginning of the source word. Transformation rules from the pair (beers, زری‌ب) are:

[1]In all the source-target instances, for clarity of alignment the Persian word is written left-to-right with characters separated.

(# , b , –) → (ب , 1.0) ,
(b , e , –) → (ى , 1.0) ,
(e , e , –) → (ε , 1.0) ,
(e , r , –) → (ر , 1.0) ,
(r , s , –) → (ز , 1.0) .

Following segmentation and transformation rule generation, the transliteration model is formed, based on which, transliteration on unseen source words can proceed.

3.2.2 Transliteration Generation

The final stage of transliteration, using the transformation rules generated in the training step, computes the probability of transliterating a source word S to a target word T as

$$P(T|S) = \prod_{i=1}^{I} P(\hat{T}_i|\hat{S}_i). \tag{3.1}$$

where I represents the number of segments generated for the source word S.

The target words can then be sorted by $P(T|S)$, given in Equation 3.1, to provide a ranked list L of the most likely transliterations of S. The selection of transformation rules to be applied requires a segmentation to be preformed on the source word, based on the segments acquired in the training stage. We therefore employ direct probability calculation similar to the general model given in Equation 2.13 on page 42.

In all methods that employ a non-empty context, provision must be made for *backoff* to a smaller context. For example, if attempting to transliterate the symbol "o" in the word "lighthouse" in the context $c_v = s_{i-4}s_{i-3}s_{i-2}s_{i-1}$ = "ghth" then it is likely that the context "ghth" occurred very infrequently in the training corpus, and so statistics derived in this context may not be reliable. In this case, a backoff scheme may try the context $c_v = s_{i-3}s_{i-2}s_{i-1}$ ="hth", which again may not occur frequently enough to be trusted, and so the next backoff context must be tried. This backoff method is used in the PPM data compression scheme [Cleary and Witten, 1984].

There are several alternate implementations for backoff schemes to apply when freq($\hat{T}|\hat{S}$) is less than a threshold (d); all are defined based on the frequency of source-target transliteration segments. We introduce two backoff methods for the context defined for $v\backslash f$. The

first approach, which we call BACKOFF-A, shrinks both past and future horizons at the same time as shown below.

$$P(\hat{T}|\hat{S}) = \begin{cases} P(\hat{T}|c_v, \hat{s}, c_f) & \text{if freq}(c_v, \hat{s}, c_f) \geq d \text{ and } v > 0 \text{ and } f > 0; \\ P(\hat{T}|c_{v-1}, \hat{s}, c_{f-1}) & \text{if freq}(c_{v-1}, \hat{s}, c_{f-1}) \geq d, \text{ freq}(c_v, \hat{s}, c_f) < d \\ & \text{and } v-1 > 0 \text{ and } f-1 > 0; \\ P(\hat{T}|\hat{s}, c_{f-1}) & \text{if freq}(\hat{s}, c_{f-1}) \geq d, \text{ freq}(c_v, \hat{s}, c_f) < d \text{ and} \\ & v = 0 \text{ and } f-1 > 0; \\ P(\hat{T}|c_{v-1}, \hat{s}) & \text{if freq}(c_{v-1}, \hat{s}) \geq d, \text{ freq}(c_v, \hat{s}, c_f) < d \text{ and} \\ & v-1 > 0 \text{ and } f = 0; \\ P(\hat{T}|\hat{s}) & \text{if } v = 0 \text{ and } f = 0. \end{cases}$$

The second approach, BACKOFF-B, first fully reduces future contexts, and only when no future context remains, begins to reduce past contexts.

Overall, the use of context has been shown to improve transliteration accuracy. In his study of transliterating six romantic languages, Lindén [2005] uses one previous source symbol and two following source symbols as context to get $P(\hat{T}|\hat{S} = s_{i-1}s_is_{i+1}s_{i+2})$, which gave an improvement of 69% over a baseline technique. In similar work on Korean, Jung et al. [2000] proposed taking advantage of past and future source symbols, and one past target language symbol, to compute $P(\hat{T}|\hat{S} = s_{i-2}s_{i-1}s_is_{i+1}, t_{i-1})$, which gave a 5% improvement over their baseline.

3.3 Implementation Description

This section provides a detailed explanation of the transliteration steps from the implementation perspective.

3.3.1 Transformation Rule Selection

If a previously unseen source word is input to the system, no information about what valid symbols it might include is available. The transliteration model formed in the training stage defines the bounds of segments for these words. Although such restrictions could disadvantage some possible transliterations, on the bright side, the restrictions favour the segmentation

saving it from infinite number of options possible for each word. Each word is therefore parsed from left-to-right (right-to-left for Persian) matching its character sequences to heads of the existing transformation rules. Lengths of these sequences are defined in accordance to the longest segment generated in the training stage. For example, in the set of segments generated using 1\0 for the example in Figure 3.2, maximum length of source segments is four because of the segments "#sch" and "scha". Each segment consists of symbol and context, and the transformation rule to be chosen should first match to the untranslated characters of the symbol, where maximum context coverage is desirable. That is, when matching the rules to the segments, the mandatory condition is that all the source characters should receive a transliteration, while optionally these transliterations are chosen based on the maximum context knowledge possible. A high-level description of segmentation of each test source word S is given in Algorithm 1.

As an example, we assume that the word "pheo" is input for transliteration and the following sample transliteration rules are available after a 1\0 model training:

(# , ph , –) → (ف , 0.5),
(# , ph , –) → (ه پ , 0.3),
(# , ph , –) → (پ , 0.2),
(ε , ph , –) → (ف , 0.3),
(ph , e , –) → (ی , 0.6),
(ph , e , –) → (ε , 0.3),
(ph , e , –) → (ا , 0.1),
(t , o , –) → (ε , 0.9),
(ε , o , –) → (و , 0.8),
(ε , o , –) → (ε , 0.2).

Segmentation in testing stage generates "ph", "e", "o" segments using the rules with {#ph, phe, o} headings. It can be noted that where proper context support is not available from the segments generated using 1\0 model (for example, for the bigram "eo"), a backoff to 0\0 is vital to cover the word thoroughly. This also applies for other similar approaches such as 2\0 which requires 0\0 and 1\0 segments as backoff.

Algorithm 1 Transformation rule selection in transliteration stage.

Require: Heads of transformation rules, (c_v, s, c_f), generated from the training stage of a $v\backslash f$ model.

Let S be the source word to be segmented.
Let $C = \#^v S \#^f$.
Let $i \rightarrow v + 1$.
while $i \leq |C|$ **do**
  Set $V \leftarrow v$, $F \leftarrow f$, and $m \leftarrow (\varepsilon,\varepsilon,\varepsilon)$.
  repeat
    if $V = 0$ and $F > 0$ **then**
      Find δ and δ_f so that $(\varepsilon,s_i \ldots s_{i+\delta-1},s_{i+\delta} \ldots s_{i+\delta+\delta_f-1})$ is longest match.
      Set $m \leftarrow (\varepsilon,s_i \ldots s_{i+\delta-1},s_{i+\delta} \ldots s_{i+\delta+\delta_f-1})$.
    else if $V > 0$ and $F = 0$ **then**
      Find δ_v and δ so that $(s_{i-\delta_v} \ldots s_{i-1},s_i \ldots s_{i+\delta-1},\varepsilon)$ is longest match.
      Set $m \leftarrow (s_{i-\delta_v} \ldots s_{i-1},s_i \ldots s_{i+\delta-1},\varepsilon)$.
    else if $V = 0$ and $F = 0$ **then**
      Find δ_v and δ so that $(\varepsilon,s_i \ldots s_{i+\delta-1},\varepsilon)$ is longest match.
      $m \leftarrow (\varepsilon,s_i \ldots s_{i+\delta-1},\varepsilon)$.
    else
      Find δ_v, δ, and δ_f so that $(s_{i-\delta_v} \ldots s_{i-1},s_i \ldots s_{i+\delta-1},s_{i+\delta} \ldots s_{i+\delta+\delta_f-1})$ is longest match.
      $m \leftarrow (s_{i-\delta_v} \ldots s_{i-1},s_i \ldots s_{i+\delta-1},s_{i+\delta} \ldots s_{i+\delta+\delta_f-1})$.
    end if
    if $m = (\varepsilon,\varepsilon,\varepsilon)$ **then**
      Backoff V or F, or both as appropriate.
    else
      Record $s_i \ldots s_{i+\delta-1}$ as a segment.
      $i \leftarrow i + \delta$.
    end if
  until $m \neq (\varepsilon,\varepsilon,\varepsilon)$
end while
Output segmented S.

3.3.2 Transliteration Variation Generation

Given the segments of a source word, transliteration proceeds using Algorithm 2 to generate the N highest scoring transliterations. This algorithm generates a tree which branches based on the source word segments. The number of segments defines the depth of the tree, and the number of siblings in each level equals to the possible transliterations of that level's segment. A schematic view of this tree is drawn in Figure 3.3 for the previous example source word "pheo" with three segments, namely "$S_1S_2S_3$". Starting from the root, one child and one sibling are added to the tree at a time while the criterion for choosing a child or sibling to be added is their probability. Therefore, the first generated leaf will have the maximum probability among all other leaves and accordingly it will be the highest probable transliteration for the source word. Note that Figure 3.3 shows the exact steps of creating the tree only until the first leaf is generated (Step I to IV); and the last tree is a view of the tree after 12 additional steps (not shown).

3.3.3 Ambiguous Alignments

Persian-English transliteration is problematic because short vowels are omitted in Persian script. Transliteration from Persian to English should embed enough vowels in the English variant to make it readable. An n-gram based transliteration which works purely based on the statistics of n-grams seen in the training data, should obtain knowledge of extra characters — to be added at transliteration generation time — using alignments. An example of alignments between Persian words and their English transliterations is shown in Figure 3.4. An English vowel "a" can be the transliteration of "ا" [ɒ] in Persian (in the second word "nazgol"), or just be added for smoother pronunciation (in the first word "nastaran" aligned with ε). To capture the rules of transliteration that lead to generating short vowels in English words, we consistently connect empty alignments to their previous character, making a multi-character symbol in the target language. For example, in the first word-pair (نسترنا, nastaran), a 0\0 model generates rules as:

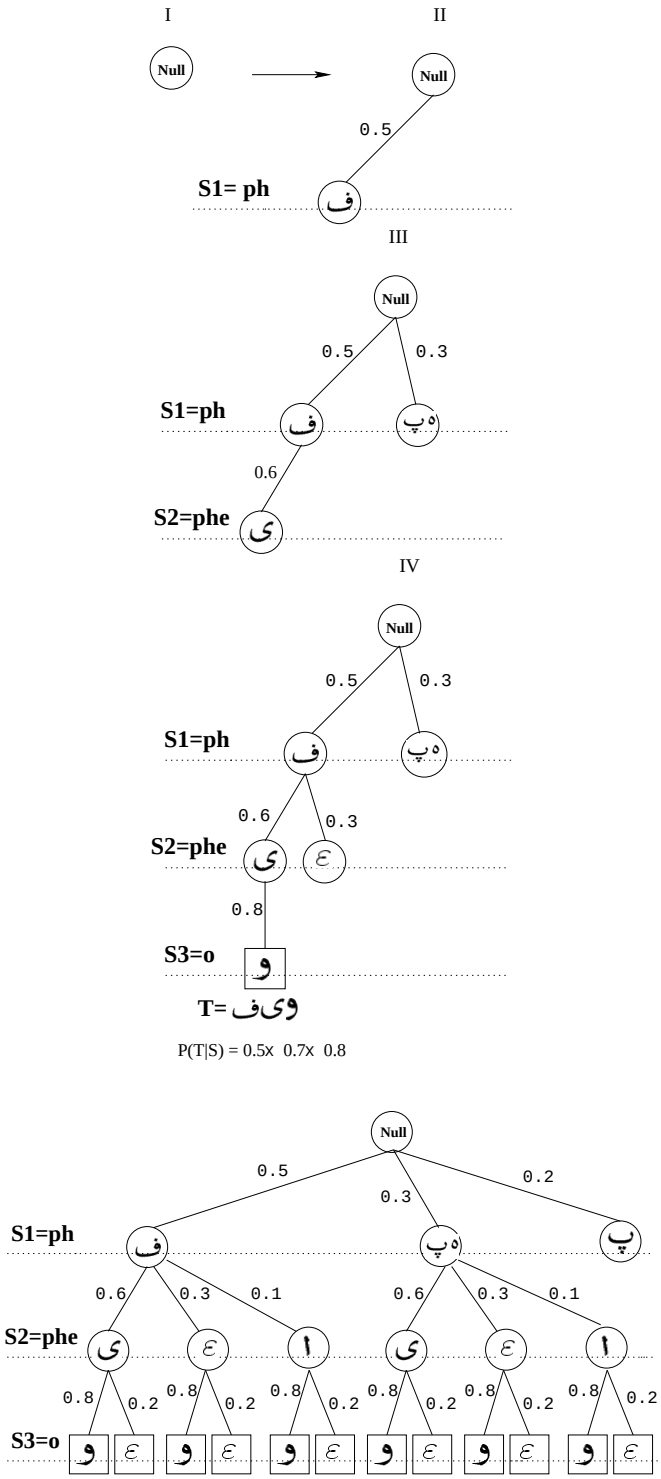


Figure 3.3: Branching of transliteration tree for an example source word S="pheo" with three segments $S_1S_2S_3$.

Algorithm 2 Transliteration output generation.

Require: Segmented source word $S = \hat{S}_1 \ldots \hat{S}_l$.

1: Let $R[\hat{S}_i]$ with $(\hat{T}_j, p_j)$ elements be a list of the transformation rules with $\hat{S}_i$ representing the source segment, $\hat{T}_j$ representing the target segment, and the list R is sorted on the p_j transformation probabilities.
2: Let Q be a priority queue initialised with root $< t = \epsilon, i = 1, j = 1, p = 1 >$; keyed on probabilities p, where t is the output string so far, i is an index into the input string S, j is an index into list $R[\hat{S}_i]$, p is probability of t occurring.
3: Let L be a list of N most probable transliterations with (t, p) elements and initialised with (ε,0).
4: Let $\min(L)$ be a function that returns the least probable element of L.
5: **while** Q is not empty **do**
6: Pop $< t, i, j, p >$ from Q.
7: **if** $p > \min(L).p$ **then**
8: **if** $i \leq \text{length}(S)$ **then**
9: Set t' appended $R[\hat{S}_i].\hat{T}_j$ to t.
10: Push $< t', i + 1, 1, p \times R[\hat{S}_i].p_j >$ into Q.
11: **else**
12: Append (t, p) to L, discarding $\min(L)$.
13: **end if**
14: **if** $j \leq length(R[\hat{S}_i])$ **then**
15: Set t' appended $R[\hat{S}_i].\hat{T}_{j+1}$ to t.
16: Push $< t', i, j + 1, p \times R[\hat{S}_i].p_{j+1} >$ into Q.
17: **end if**
18: **end if**
19: **end while**
20: Output L.

Training set:	**Alignment:**							
	ن	ε	س	ت	ε	ر	ε	ن
(نسترن, nastaran)	|	|	|	|	|	|	|	|
	n	a	s	t	a	r	a	n
	ن	ا	ز	گ	ε	ل		
(نازگل, nazgol)	|	|	|	|	|	|		
	n	a	z	g	o	l		
		ش		ی	ر	ا	ز	
(شیراز, shiraz)	/		\	|	|	|	|	
	s		h	i	r	a	z	

Figure 3.4: Sample word-pairs and their alignments from a Persian-English transliteration process. Short vowels that did not have a counterpart in Persian source word, are aligned with empty string (ε).

$(-, \text{ن}, -) \rightarrow (\text{na}, 0.33)$,
$(-, \text{س}, -) \rightarrow (\text{s}, 1.0)$,
$(-, \text{ت}, -) \rightarrow (\text{ta}, 1.0)$,
$(-, \text{ر}, -) \rightarrow (\text{ra}, 0.5)$,
$(-, \text{ن}, -) \rightarrow (\text{n}, 0.66)$.

3.4 Experimental Setup

We conduct experiments varying past and future context sizes for both English-Persian and Persian-English transliteration. The corpora used for English-Persian transliteration are B^{+}_{e2p} and B_{e2p} (the first contains 16,670 and the second contains 1,870 word-pairs — see Section 2.2.3 for details), and for Persian-English transliteration B_{p2e} is used (contains 2,010 word-pairs). The results are reported using both word accuracy and character accuracy metrics (see Section 2.2.4) in TOP-1, TOP-5, TOP-10, and TOP-20. Since B_{p2e} contains more than one transliteration variant for each source word, we value each of these equally in our

evaluations[2].

Alignment of source and target words, for both English-Persian and Persian-English, is performed using GIZA++ [3] from which we obtain symbols, and subsequently use n-gram models to form the transformation rules.

The target language model used in our experiments is a first order language model based on bigrams, applied to the output list of each system to re-rank the generated transliterations.

3.5 Past Context

In the first set of experiments, the effect of using only past context is studied. We examine a context of at most two past symbols, and therefore 0\0, 1\0, and 2\0 methods are considered.

The results of applying 0\0, 1\0 and 2\0 methods (employing no context, one, and two past symbols) respectively, and evaluated based on word accuracy metric, are shown in Table 3.1. For English-Persian transliteration using B_{e2p} corpus, the 0\0 model performed equal to 2\0, while 1\0 performed significantly better than 0\0 only for TOP-1 (paired t-test, $p = 0.00234$). Comparing 1\0 and 2\0 methods, 1\0 was significantly better from TOP-1 to TOP-20 results. Using the B^{+}_{e2p} corpus, 1\0 outperformed both other methods from TOP-1 to TOP-20, while using larger context performed worse than only one past symbol. Therefore, using past context greater than only one symbol did not assist the transliteration.

In Persian-English transliteration, in TOP-1 to TOP-20 results, 1\0 outperformed both other methods significantly (paired t-test, $p < 0.0001$). There was no significant difference between 0\0 and 2\0 in TOP-1, but 0\0 was more effective in TOP-5 to TOP-20 results.

Table 3.2 compares the three methods using the mean character accuracy metric for TOP-1 transliterations only. In terms of character accuracy, in English-Persian transliteration there is no difference between the three methods applying past context evaluated on the B_{e2p} corpus. For the B^{+}_{e2p} corpus, however, 1\0 is significantly better than both the 0\0 and 2\0 approaches. In Persian-English transliteration also, the character accuracy metric evaluates 1\0 significantly higher than both 0\0 and 2\0 methods.

The number of distinct segments for source and target along with their combinations as transformation rules generated for each training corpus is reported in Table 3.3. According

[2] This metric is called UWA (uniform word accuracy). Please refer to Section 2.5.4 on page 62 for details.

[3] GIZA++ is described in detail in Section 2.1.3, page 29.

to this table, in English-Persian transliteration the number of source segments increases dramatically with context, particularly using the B^{+}_{e2p} corpus. The number of target segments is noticeably smaller than the number of source segments.

We repeated the experiments to evaluate the use of target language models, introducing the following systems: 0\0T, 1\0T, and 2\0T; results are reported in Table 3.4. Accuracy obtained using a target language model is noticeably lower than the same systems without applying a target language model on their output. In English-Persian transliteration, 1\0T is more accurate than 0\0T and 2\0T evaluating on both B_{e2p} and B^{+}_{e2p}. Transliterating from Persian to English was an obvious failure for 0\0T with mean word accuracy less than 1% TOP-1. Two other systems, 1\0T and 2\0T, performed similarly, but worse than 1\0 and 2\0 methods that do not apply target language model.

In total, the highest word accuracy was obtained using only past context was 58.0% TOP-1 for English to Persian transliteration, and 33.9% TOP-1 for Persian to English. Both of these are for the 1\0 approach. The character accuracy metric confirms the superiority of the 1\0 approach over other methods with a maximum of 89.2% TOP-1 for English-Persian and 80.5% TOP-1 for Persian-English. We therefore use 1\0, as our best system so far, as a baseline for comparisons in the next sets of experiments reported below.

3.6 Future Context

The effect of using future context is studied in our second set of experiments (Table 3.5). Here again, we expand the context to a maximum of two symbols, therefore 0\1 and 0\2 methods are examined. Comparisons are made against 1\0, the best performing system from Table 3.1. Systems which use future context only obtained different performances for evaluations on B_{e2p} and B^{+}_{e2p} corpora using the word accuracy metric. Using B_{e2p}, all three methods, 1\0, 0\1, and 0\2, perform equally for TOP-1, while 0\2 performs worse than others for TOP-5 to TOP-20, and 0\1 and 0\2 do not show significant difference. On the B^{+}_{e2p} corpus, using two future context (0\2) significantly ($p< 0.0001$) outperforms the 0\1 method that uses only one future symbol.

Persian to English transliteration does not differentiate between 0\1 and 0\2 in TOP-1 results, giving a 25.4% accuracy for both. In TOP-5 to TOP-10 results, the 0\2 method is the worst system, and 1\0 and 0\1 compete closely with no significant difference; whereas in

		Method			t-test p-value		
Corpus		0\0	1\0	2\0	(0\0,1\0)	(0\0,2\0)	(1\0,2\0)
B_{e2p}	TOP-1	46.6 (19.7)	58.0 (3.4)	44.6 (14.6)	0.0234	0.4117	0.0013
	TOP-5	72.0 (19.5)	85.6 (2.2)	69.1 (14.6)	0.0532	0.4902	0.0048
	TOP-10	78.4 (21.6)	89.4 (2.9)	72.6 (15.7)	0.1291	0.2091	0.0049
	TOP-20	82.0 (22.3)	90.5 (3.1)	73.9 (16.1)	0.2372	0.1057	0.0059
B^{+}_{e2p}	TOP-1	18.4 (9.5)	47.2 (1.0)	11.0 (6.3)	< 0.0001	0.0546	< 0.0001
	TOP-5	28.7 (15.2)	77.6 (1.4)	18.3 (11.2)	< 0.0001	0.0849	< 0.0001
	TOP-10	31.2 (16.6)	83.3 (1.5)	20.1 (13.1)	< 0.0001	0.0948	< 0.0001
	TOP-20	32.5 (17.2)	86.1 (1.4)	21.0 (13.1)	< 0.0001	0.0946	< 0.0001
B_{p2e}	TOP-1	13.6 (1.3)	33.9 (3.8)	12.4 (2.8)	< 0.0001	0.3051	< 0.0001
	TOP-5	36.2 (3.2)	51.6 (2.8)	32.5 (4.2)	< 0.0001	0.0371	< 0.0001
	TOP-10	46.2 (2.9)	56.1 (2.6)	37.0 (3.6)	< 0.0001	0.0002	< 0.0001
	TOP-20	55.0 (2.9)	58.8 (2.9)	40.4 (3.1)	0.0034	< 0.0001	< 0.0001

Table 3.1: Mean word accuracy (as percentages) of English-Persian and Persian-English transliterations, changing past context sizes. Standard deviations are given in parentheses. English-Persian is evaluated on B_{e2p} and B^{+}_{e2p}, and Persian-English on B_{p2e}.

	Method			t-test p-value		
Corpus	0\0	1\0	2\0	(0\0,1\0)	(0\0,2\0)	(1\0,2\0)
B_{e2p}	80.3 (15.6)	89.2 (1.0)	81.9 (12.7)	0.0993	0.6248	0.0927
B^{+}_{e2p}	56.7 (16.4)	85.5 (0.5)	43.6 (10.8)	0.0004	0.0381	< 0.0001
B_{p2e}	73.6 (1.1)	80.5 (1.1)	68.5 (1.7)	< 0.0001	< 0.0001	< 0.0001

Table 3.2: Mean character accuracy (TOP-1) *for English-Persian and Persian-English transliteration. Only past context is used.*

	Source			Target			Source-Target		
Corpus	0\0	1\0	2\0	0\0	1\0	2\0	0\0	1\0	2\0
B_{e2p}	22	61	75	27	49	59	50	152	180
B^{+}_{e2p}	103	1,199	7,989	41	112	199	162	10,422	19,015
B_{p2e}	37	526	2,023	119	207	708	176	982	2,081

Table 3.3: Number of segments for changing past context sizes for English-Persian and Persian-English transliteration.

TOP-20, 0\1 acquires the highest accuracy of 65.6%.

Table 3.6 compares the two 0\1 and 0\2 systems and the baseline using mean character accuracy in TOP-1 transliterations. For Persian-English transliteration, there is no difference between the three methods, and the same applies to English-Persian transliteration on the B_{e2p} corpus. However, when English-Persian transliteration is evaluated on B^{+}_{e2p}, 1\0 significantly outperforms 0\1 and 0\2 with 85.5% mean character accuracy.

The statistics of source and target segments generated by each method for both B_{e2p} and B^{+}_{e2p} corpora are given in Table 3.7. The number of English segments generated is more than Persian segments in both English-Persian and Persian-English transliterations.

Experiments using the mean word accuracy metric on systems that apply target language model, that is 0\1T and 0\2T, and reported in Table 3.8. Using a target language model did not help to improve transliteration with significantly degrading the accuracy for both English-Persian and Persian-English transliterations.

According to the results obtained from both sets of experiments using either past or future context, we can draw the conclusion that 1\0 performs reliably better than other variants. That is, even if equal word or character accuracy is obtained using 1\0 in comparison to others, it does not fall behind any other method based on the reported experiments.

3.7 Past and Future Context: Symmetrical and Non-Symmetrical

In this section, we explore the effect of using both past and future contexts together with the influence of backoff methods. The results of using symmetrical context, 1\1 and 2\2, is shown in Table 3.9 with comparisons made against 1\0, the best method from the previous

		Method			t-test p-value		
Corpus		0\0T	1\0T	2\0T	(0\0T,1\0T)	(0\0T,2\0T)	(1\0T,2\0T)
B_{e2p}	TOP-1	31.8 (11.2)	55.0 (3.4)	42.8 (10.7)	0.0002	0.0010	0.0075
	TOP-5	42.7 (13.2)	78.9 (2.2)	65.4 (14.7)	< 0.0001	< 0.0001	0.0156
	TOP-10	48.7 (14.1)	82.7 (1.8)	70.3 (15.2)	< 0.0001	0.0001	0.0258
	TOP-20	60.2 (15.6)	86.0 (2.8)	72.8 (15.8)	0.0004	0.0052	0.0199
B^{+}_{e2p}	TOP-1	4.4 (3.5)	41.3 (1.3)	3.7 (2.2)	< 0.0001	0.6187	< 0.0001
	TOP-5	6.2 (5.0)	64.0 (1.4)	5.1 (3.4)	< 0.0001	0.6374	< 0.0001
	TOP-10	7.1 (5.9)	69.5 (1.6)	5.9 (3.8)	< 0.0001	0.6408	< 0.0001
	TOP-20	9.3 (7.6)	74.0 (1.6)	6.6 (4.3)	< 0.0001	0.3953	< 0.0001
B_{p2e}	TOP-1	0.6 (0.8)	21.9 (0.9)	20.5 (3.9)	< 0.0001	< 0.0001	0.3079
	TOP-5	0.8 (1.0)	41.0 (2.8)	33.4 (3.6)	< 0.0001	< 0.0001	0.0025
	TOP-10	1.0 (1.1)	48.5 (2.6)	38.0 (4.3)	< 0.0001	< 0.0001	0.0006
	TOP-20	1.1 (1.1)	54.8 (2.9)	41.3 (3.8)	< 0.0001	< 0.0001	< 0.0001

Table 3.4: Mean word accuracy (as percentages) of English-Persian and Persian-English transliterations changing past context sizes and using target language model. Standard deviations are given in parentheses. English-Persian is evaluated on B_{e2p} and B^{+}_{e2p}, and Persian-English is evaluated on B_{p2e}.

experiments.

Symmetrical context does not help English to Persian transliteration in comparison to using only one past symbol context. As can be seen in Table 3.9, for B_{e2p}, the three methods are not statistically significantly different in TOP-1, while from TOP-5 to TOP-20 adding more context reduces the word accuracy. Experiments with B^{+}_{e2p} which show similar results, with TOP-1 significantly worse for symmetrical methods in comparison to 1\0. Transliteration from Persian to English, in contrast, is more accurately achieved using symmetric context giving 22.1% and 24.5% relative improvement over 1\0 for 1\1 and 2\2 approaches, respectively. These methods themselves are not significantly different in TOP-1, but 1\1 outperforms 2\2 for TOP-5 to TOP-20 results.

Symmetric context systems with a target language model are examined and reported

Corpus		Method			t-test p-value		
		1\0	0\1	0\2	(1\0,0\1)	(1\0,0\2)	(0\1,0\2)
B_{e2p}	TOP-1	58.0 (3.4)	53.1 (12.5)	54.6 (11.9)	0.2416	0.1341	0.0932
	TOP-5	85.6 (2.2)	78.3 (16.4)	77.0 (15.3)	0.1621	0.0122	< 0.0001
	TOP-10	89.4 (2.9)	80.6 (16.6)	79.4 (15.2)	0.1034	0.0040	< 0.0001
	TOP-20	90.5 (3.1)	81.2 (16.6)	80.3 (15.1)	0.0899	0.0028	< 0.0001
B^{+}_{e2p}	TOP-1	47.2 (1.0)	9.8 (11.6)	23.1 (13.8)	< 0.0001	0.0004	< 0.0001
	TOP-5	77.6 (1.4)	17.5 (21.5)	32.2 (25.5)	< 0.0001	0.0004	< 0.0001
	TOP-10	83.3 (1.5)	19.4 (23.7)	41.3 (24.8)	< 0.0001	0.0003	0.0001
	TOP-20	86.1 (1.4)	20.5 (25.1)	42.1 (25.2)	< 0.0001	0.0003	0.0001
B_{p2e}	TOP-1	33.9 (3.8)	25.4 (1.9)	25.4 (2.2)	0.0002	< 0.0001	0.5585
	TOP-5	51.6 (2.8)	50.3 (3.2)	45.3 (2.0)	0.4230	0.0011	< 0.0001
	TOP-10	56.1 (2.6)	59.5 (3.2)	51.4 (2.3)	0.0634	0.0171	< 0.0001
	TOP-20	58.8 (2.9)	65.6 (3.7)	55.5 (2.2)	0.0039	0.0589	< 0.0001

Table 3.5: Mean word accuracy (as percentages) in English-Persian and Persian-English transliteration for changing future context sizes. Standard deviations are given in the parentheses.

Corpus	Method			t-test p-value		
	1\0	0\1	0\2	(1\0,0\1)	(1\0,0\2)	(0\1,0\2)
B_{e2p}	89.2 (1.0)	84.7 (13.6)	84.5 (13.5)	0.3048	0.2795	0.3326
B^{+}_{e2p}	85.5 (0.5)	39.3 (19.6)	58.2 (19.3)	< 0.0001	0.0015	< 0.0001
B_{p2e}	80.5 (1.1)	79.9 (1.2)	80.2 (1.2)	0.2929	0.5437	0.2065

Table 3.6: Mean character accuracy (TOP-1) *for English-Persian and Persian-English transliteration, for changing future context sizes. Standard deviations are given in the parentheses.*

	Source		Target		Source-Target	
Corpus	0\1	0\2	0\1	0\2	0\1	0\2
B_{e2p}	563	2,013	57	83	749	2,304
B^{+}_{e2p}	1,229	6,814	189	113	1,964	8,864
B_{p2e}	321	481	468	1,360	1,112	2,446

Table 3.7: Segment size for changing future context sizes.

		Method			t-test p-value		
Corpus		1\0	0\1T	0\2T	(1\0,0\1T)	(1\0,0\2T)	(0\1T,0\2T)
B_{e2p}	TOP-1	58.0 (3.4)	38.6 (8.8)	49.8 (11.4)	< 0.0001	0.0438	< 0.0001
	TOP-5	85.6 (2.2)	66.5 (14.1)	68.8 (14.9)	0.0013	0.0040	0.0418
	TOP-10	89.4 (2.9)	75.2 (16.2)	71.7 (15.1)	0.0154	0.0031	0.0009
	TOP-20	90.5 (3.1)	79.8 (16.9)	72.5 (15.1)	0.0602	0.0027	< 0.0001
B^{+}_{e2p}	TOP-1	47.2 (1.0)	6.9 (6.5)	14.5 (9.9)	< 0.0001	< 0.0001	0.0108
	TOP-5	77.6 (1.4)	12.9 (13.1)	27.2 (18.8)	< 0.0001	< 0.0001	0.0102
	TOP-10	83.3 (1.5)	16.5 (17.0)	33.2 (23.3)	< 0.0001	< 0.0001	0.0129
	TOP-20	86.1 (1.4)	19.4 (20.3)	36.7 (25.5)	< 0.0001	0.0002	0.0173
B_{p2e}	TOP-1	33.9 (3.8)	17.2 (2.7)	21.3 (3.4)	< 0.0001	< 0.0001	0.0004
	TOP-5	51.6 (2.8)	33.1 (3.4)	36.7 (4.2)	< 0.0001	< 0.0001	0.0167
	TOP-10	56.1 (2.6)	46.0 (4.9)	45.3 (2.6)	0.0002	< 0.0001	0.5308
	TOP-20	58.8 (2.9)	56.1 (4.6)	51.5 (2.0)	0.2339	0.0004	0.0024

Table 3.8: Mean word accuracy (as percentages) in English-Persian and Persian-English transliteration for changing future context sizes using target language model (except baseline 1\0*). Standard deviations are given in the parentheses.*

		Method			t-test p-value		
Corpus		1\0	1\1	2\2	(1\0, 1\1)	(1\0, 2\2)	(1\1, 2\2)
B_{e2p}	TOP-1	58.0 (3.4)	56.7 (14.2)	57.3 (15.1)	0.7814	0.8939	0.3075
	TOP-5	85.6 (2.2)	73.5 (17.2)	70.7 (17.2)	0.0492	0.0204	< 0.0001
	TOP-10	89.4 (2.9)	75.4 (17.8)	72.3 (17.7)	0.0248	0.0091	0.0001
	TOP-20	90.5 (3.1)	76.0 (17.8)	72.7 (17.7)	0.0206	0.0070	< 0.0001
B^{+}_{e2p}	TOP-1	47.2 (1.0)	40.7 (1.5)	39.5 (1.2)	< 0.0001	< 0.0001	< 0.0001
	TOP-5	77.6 (1.4)	63.2 (2.4)	55.7 (1.9)	< 0.0001	< 0.0001	< 0.0001
	TOP-10	83.3 (1.5)	66.7 (2.6)	57.5 (2.6)	< 0.0001	< 0.0001	< 0.0001
	TOP-20	86.1 (1.4)	68.0 (2.6)	58.2 (2.6)	< 0.0001	< 0.0001	< 0.0001
B_{p2e}	TOP-1	33.9 (3.8)	41.4 (2.6)	42.2 (2.5)	0.0004	0.0002	0.0699
	TOP-5	51.6 (2.8)	63.2 (3.3)	61.2 (3.5)	< 0.0001	< 0.0001	< 0.0001
	TOP-10	56.1 (2.6)	68.2 (2.6)	65.3 (2.9)	< 0.0001	< 0.0001	< 0.0001
	TOP-20	58.8 (2.9)	71.3 (2.8)	67.7 (2.6)	< 0.0001	< 0.0001	< 0.0001

Table 3.9: Mean word accuracy (as percentages) of English-Persian and Persian-English transliteration for symmetric past and future context size with BACKOFF-A. *Standard deviations are given in parentheses.*

in Table 3.12. English to Persian transliteration does not improve using our bigram target language model, neither on B_{e2p} nor on B^{+}_{e2p}. Surprisingly however, Persian to English transliteration using the 2\2T approach benefits from a target language model in comparison to the 2\2 method showing 47.0% TOP-1 accuracy versus 42.2% word accuracy, this translates to a significant relative improvement of 11.4% in TOP-1 over the 2\2 method ($p< 0.0001$).

Comparing the systems using the mean character accuracy measure in TOP-1 transliterations resembles the same results of word accuracy for English-Persian and Persian-English, as shown in Table 3.10. English-Persian benefits from 1\0 context more than from using symmetric context, whereas Persian-English prefers symmetric past and future context over just one past context, giving a maximum of 87.7% character accuracy using the 1\1 method, which translates to a relative improvement of 8.9%.

Non-symmetric context with two different backoff methods are also investigated and re-

	Method			t-test p-value		
Corpus	1\0	1\1	2\2	(1\0,1\1)	(1\0,2\2)	(1\1,2\2)
B_{e2p}	89.2 (1.0)	85.7 (13.8)	85.8 (14.2)	0.4334	0.4557	0.5831
B^{+}_{e2p}	85.5 (0.5)	81.9 (1.5)	81.5 (1.5)	< 0.0001	< 0.0001	< 0.0001
B_{p2e}	80.5 (1.1)	87.7 (1.3)	87.6 (1.2)	< 0.0001	< 0.0001	0.6020

Table 3.10: Mean character accuracy (TOP-1) *of English-Persian and Persian-English transliteration for changing symmetric past and future context size.*

	Source		Target		Source-Target	
Corpus	1\1	2\2	1\1	2\2	1\1	2\2
B_{e2p}	2,570	9,037	66	101	2,802	9,350
B^{+}_{e2p}	8,701	44,759	141	243	11,292	58,542
B_{p2e}	327	523	2,296	7,998	2,948	6,317

Table 3.11: Number of segments for changing symmetric past and future context size for English-Persian and Persian-English transliteration.

ported in Table 3.13. Using two past and one future context has a negative impact on perceived accuracy for both English-Persian and Persian-English. Comparing the two approaches of 2\1A (2\1 method that uses BACKOFF-A) and 2\1B (2\1 method that uses BACKOFF-B), they perform equally for English-Persian using B_{e2p}, but 2\1A performs more effectively than 2\1B when compared on the B^{+}_{e2p} or B_{p2e} corpora.

Character accuracy, shown in Table 3.14, confirms the conclusions drawn based on the word accuracy metric. English-Persian is more accurately performed using 1\0 in comparison to 2\1A and 2\1B. For Persian-English transliteration, however, there is no difference between 1\0 and 2\1A, with 2\1B performing the worst.

When a target language model is used with the 2\1A and 2\1B methods (Table 3.16), it harms the results in comparison to original methods without a target language model and also in comparison to the baseline 1\0 approach.

The number of segments for all the approaches examined in this section is reported in Tables 3.11 and 3.15 for symmetric and non-symmetric methods, respectively. The number of source segments generated for 2\2 is remarkably high, and almost 184 times more than

		Method			t-test p-value		
Corpus		1\0	1\1T	2\2T	(1\0,1\1T)	(1\0,2\2T)	(1\1T,2\2T)
B_{e2p}	TOP-1	58.0 (3.4)	49.6 (3.3)	14.3 (4.9)	0.0007	< 0.0001	< 0.0001
	TOP-5	85.6 (2.2)	74.7 (2.6)	29.0 (9.8)	< 0.0001	< 0.0001	< 0.0001
	TOP-10	89.4 (2.9)	81.8 (2.5)	33.7 (11.7)	0.0104	< 0.0001	< 0.0001
	TOP-20	90.5 (3.1)	86.9 (3.1)	38.3 (13.4)	0.1169	< 0.0001	< 0.0001
B^{+}_{e2p}	TOP-1	47.2 (1.0)	15.9 (6.1)	0.7 (0.5)	< 0.0001	< 0.0001	< 0.0001
	TOP-5	77.6 (1.4)	27.7 (10.8)	1.4 (0.8)	< 0.0001	< 0.0001	< 0.0001
	TOP-10	83.3 (1.5)	38.5 (14.3)	2.0 (1.1)	< 0.0001	< 0.0001	< 0.0001
	TOP-20	86.1 (1.4)	50.4 (18.8)	2.8 (1.6)	< 0.0001	< 0.0001	< 0.0001
B_{p2e}	TOP-1	33.9 (3.8)	38.8 (3.2)	47.0 (2.4)	0.0101	< 0.0001	0.0007
	TOP-5	51.6 (2.8)	64.4 (3.7)	66.5 (3.1)	< 0.0001	< 0.0001	0.2310
	TOP-10	56.1 (2.6)	70.5 (3.6)	70.3 (3.4)	< 0.0001	< 0.0001	0.9115
	TOP-20	58.8 (2.9)	74.2 (2.8)	72.7 (2.8)	< 0.0001	< 0.0001	0.3279

Table 3.12: Mean word accuracy (as percentages) of English-Persian and Persian-English transliteration for symmetric past and future context size with BACKOFF-A *and using a target language model. Standard deviations are given in parentheses.*

		Method			t-test p-value		
Corpus		1\0	2\1A	2\1B	(1\0,2\1A)	(1\0,2\1B)	(2\1A,2\1B)
B_{e2p}	TOP-1	58.0 (3.4)	43.1 (10.9)	40.2 (10.4)	0.0024	0.0006	0.3414
	TOP-5	85.6 (2.2)	70.8 (16.6)	61.2 (16.0)	0.0149	0.0009	0.0557
	TOP-10	89.4 (2.9)	77.3 (18.7)	68.2 (18.5)	0.0490	0.0048	0.1106
	TOP-20	90.5 (3.1)	80.8 (20.0)	71.7 (19.3)	0.1225	0.0105	0.1083
B^{+}_{e2p}	TOP-1	47.2 (1.0)	23.9 (12.9)	10.7 (6.6)	0.0003	< 0.0001	0.0096
	TOP-5	77.6 (1.4)	41.8 (22.2)	18.3 (11.5)	0.0007	< 0.0001	0.0079
	TOP-10	83.3 (1.5)	47.0 (25.2)	21.1 (13.1)	0.0014	< 0.0001	0.0088
	TOP-20	86.1 (1.4)	49.8 (26.8)	23.7 (14.7)	0.0022	< 0.0001	0.0131
B_{p2e}	TOP-1	33.9 (3.8)	24.2 (2.7)	12.8 (2.8)	< 0.0001	< 0.0001	< 0.0001
	TOP-5	51.6 (2.8)	51.4 (2.5)	38.1 (4.2)	0.6356	< 0.0001	< 0.0001
	TOP-10	56.1 (2.6)	63.4 (1.8)	44.2 (4.0)	< 0.0001	0.0002	< 0.0001
	TOP-20	58.8 (2.9)	72.8 (3.3)	48.1 (3.6)	< 0.0001	0.0002	< 0.0001

Table 3.13: Mean word accuracy (as percentages) of English-Persian and Persian-English transliteration for changing past and future context size. Standard deviations are given in parentheses.

corresponding target segments. As per previous methods, more segments are generated for English than Persian either it is source language or target language.

The experiments reported in this section suggest that English-Persian transliteration is still best achieved using only one past context symbol and no target language model. For Persian-English, however, the symmetric method of 2\2T which uses two past and a future context symbols and target language model seems superior.

3.8 Conclusions

The first step of finding an effective method of transliteration between English and Persian is presented in this chapter where – following the general trend of past studies – we explore many possible n-gram based approaches, some of which were already attempted in the literature

	Method			t-test p-value		
Corpus	1\0	2\1A	2\1B	(1\0,2\1A)	(1\0,2\1B)	(2\1A,2\1B)
B_{e2p}	89.2 (1.0)	79.5 (14.2)	77.5 (15.0)	0.0523	0.0352	0.6280
B^{+}_{e2p}	85.5 (0.5)	60.1 (17.7)	42.0 (11.1)	0.0015	< 0.0001	0.0108
B_{p2e}	80.5 (1.1)	79.8 (1.0)	73.0 (1.1)	0.0604	< 0.0001	< 0.0001

Table 3.14: Mean character accuracy (TOP-1) *for changing past and future context size in English-Persian and Persian-English transliteration.*

	Source		Target		Source-Target	
Corpus	1\2A	1\2B	1\2A	1\2B	1\2A	1\2B
B_{e2p}	6,591	4,045	94	63	6,980	4,245
B^{+}_{e2p}	22,946	23,385	228	184	25,451	25,964
B_{p2e}	502	704	5,842	5,476	3,923	3,953

Table 3.15: Segment size for non-symmetric changing of past and future context size for English-Persian and Persian-English transliteration.

for other language pairs.

The results for transliteration of out-of-dictionary words from English to Persian using n-gram based systems show that 1\0 — using a history of just one symbol — achieves the highest word accuracy in comparison to other models. Extending the historical context causes performance to deteriorate compared to using just one past symbol. Similarly, adding future context causes performance to fall when past context is also present. Using the symmetrical backoff method, BACKOFF-A, rather than BACKOFF-B, improves performance for the larger B^{+}_{e2p} corpus when both past and future context are used in the transliteration method.

The outcomes of transliteration from Persian to English show that using symmetrical context serves Persian-English transliteration the best in comparison to other approaches we examined. Target language models, while not improving any of the English to Persian transliteration approaches, increased the accuracy of methods using symmetrical context for Persian-English, making 2\2T the optimum approach. In the competition of n-gram approaches, 1\0 was the runner-up method after 2\2T.

We therefore, by examining the role source past and future context sizes, and target

		Method			t-test p-value		
Corpus		1\0	2\1A	2\1B	(1\0,2\1A)	(1\0,2\1B)	(2\1A,2\1B)
B_{e2p}	TOP-1	58.0 (3.4)	44.0 (8.4)	16.3 (8.9)	0.0005	< 0.0001	< 0.0001
	TOP-5	85.6 (2.2)	72.2 (11.0)	28.6 (15.0)	0.0039	< 0.0001	< 0.0001
	TOP-10	89.4 (2.9)	78.6 (12.0)	20.1 (17.0)	0.0171	< 0.0001	< 0.0001
	TOP-20	90.5 (3.1)	82.0 (12.3)	28.6 (10.9)	0.0504	< 0.0001	< 0.0001
B^{+}_{e2p}	TOP-1	47.2 (1.0)	41.2 (11.9)	12.8 (9.3)	0.0018	< 0.0001	0.0003
	TOP-5	77.6 (1.4)	68.0 (17.7)	25.3 (18.7)	0.1126	< 0.0001	0.0005
	TOP-10	83.3 (1.5)	74.1 (19.8)	24.3 (18.7)	0.0266	< 0.0001	0.0004
	TOP-20	86.1 (1.4)	77.5 (21.4)	30.2 (13.9)	0.2260	< 0.0001	0.0002
B_{p2e}	TOP-1	33.9 (3.8)	24.2 (2.7)	12.8 (2.8)	< 0.0001	< 0.0001	< 0.0001
	TOP-5	51.6 (2.8)	51.4 (2.4)	38.1 (4.2)	0.6932	< 0.0001	< 0.0001
	TOP-10	56.1 (2.6)	63.4 (1.8)	44.2 (4.0)	< 0.0001	0.0002	< 0.0001
	TOP-20	58.8 (2.9)	72.8 (3.3)	48.1 (3.6)	< 0.0001	0.0002	< 0.0001

Table 3.16: Mean word accuracy (as percentages) of English-Persian and Persian-English transliteration for non-symmetric changing of past and future context size using a target language model (except baseline 1\0*). Standard deviations are given in parentheses.*

language on transliteration performance for the language pair English and Persian, found the existing transliteration methods that are more effective. This process also led to modifications required to these existing approaches to improve the effectiveness of English-Persian and Persian-English transliterations.

Transliteration approaches in this chapter were presented with the assumption of reliability of the alignment tool we used: GIZA++. In practice, however, we faced problems when parsing this tool's output to symbols and rules. The ambiguity that a language with Persian script with no short vowel imposes on parsing made the process even harder. Non-ideal alignment may lead to a reduction in transliteration efficacy. We address the impact of alignment on English to Persian transliteration effectiveness in Chapter 4.

One drawback of n-gram based transliteration approaches is their blind segmentation in training and testing stages, with no sense of the word under segmentation. Languages involved are therefore ignored together with their specific properties. To increase transliteration accuracy we should consider parameters of language features. We investigate the question of what language features may help to increase the transliteration accuracy in Chapter 4.

Chapter 4

Transliteration Based on Consonant-Vowel Sequences

"We cannot solve our current problem with the same thinking we used to create them."

— Albert Einstein

Novel transliteration approaches are presented in this chapter by introducing three transliteration methods — named CV-MODEL1, CV-MODEL2, and CV-MODEL3— that incorporate shallow but useful language features to improve translation accuracy. Specifically, the nature of letters constructing a word — being a consonant or a vowel — and also their order are considered in the transliteration process. We also investigate the idea of applying consonant and vowel knowledge in the alignment phase to gain less noisy, and more accurate transformation rules. Back-transliteration is also studied, proposing a novel algorithm to back-transliterate Persian to English.

4.1 Introduction

Purely selecting contexts according to a $v\backslash f$ model, as described in Chapter 3 on n-gram approaches, ignores any characteristics that we may know of the source and target languages. For instance, Persian speakers tend to transliterate diphthongs of other languages to monophthongs. In addition, short vowel sounds in English are often omitted altogether in written

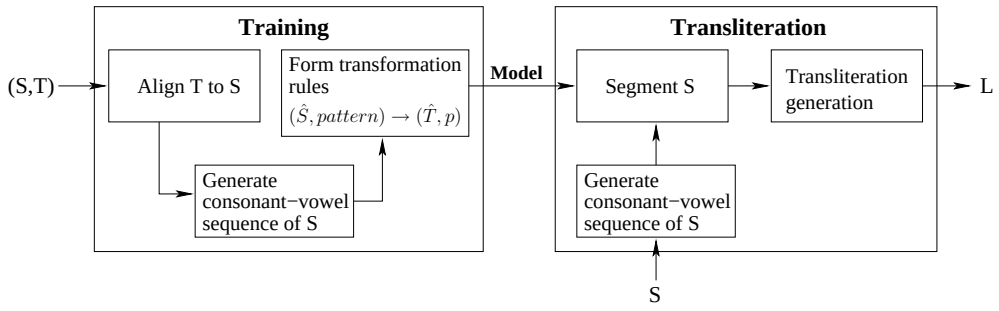


Figure 4.1: A general transliteration diagram based on consonants and vowels.

Persian. This suggests an approach where vowels are transliterated in the context of their surrounding consonants. Rather than employing a full natural language analysis technique, as has been done for Chinese [Wan and Verspoor, 1998] where syllabification of words is based on their phonetical characteristics, we explore a simpler segmentation approach which turns out to be effective.

Similar to approaches based on n-grams, two main steps are defined for our consonant-vowel based transliteration methods: training and transliteration. The sub-tasks of each of these steps are shown in Figure 4.1. The training stage includes alignment of source and target words, segmentation of the aligned pairs using defined patterns of the source word, and generation of transformation rules. The transliteration stage consists of segmentation of the source word and transliteration generation. The difference between the methods based on consonant-vowel sequences and n-grams, however, lies in their segmentation steps where the new methods avoid a fixed length segmentation approach practised as n-grams.

The ideal transliteration approach should perform well in modelling the source to target string mapping, and this task is heavily reliant on how the words are segmented. There are two reasons for this: firstly, probabilities are calculated based on these segments, and if improper segments are chosen, wrong probabilities affect the final ranking of the transliterations generated. Secondly, since two segmentation steps (training and transliteration) work based on a common rule, the former segmentation step should generate best matching segments to existing segments (registered in the heads of transformation rules), otherwise we will need to match them by applying backoff rules which may affect the consistency of the procedures. Meaning that, best matching happens when a segment of a test word is found

in the *corresponding segments* from the training data without altering its context. By corresponding segments we mean those segments created with the current segmentation approach. That is, if we are applying $1\backslash 0$ context, then in the transliteration stage search for each of the segments should be successful in transformation rules generated by $1\backslash 0$ segmentation, not its backoff supply from $0\backslash 0$. To clarify, assume that at transliteration time a segment is generated as "oph" which is missing in the heads of transformation rules. On the other hand, we know that "ph" represents one character in the target language. A backoff method normally breaks this segment to smaller segments; however, since no language information is available using $v\backslash f$ methods, there is no guarantee to re-segment "oph" to "o" and "ph". It is very likely to obtain "op" and "h" instead, parsing that from left to right. In the context of English to Persian transliteration, this example can be translated to this story that we expect the system to transliterate "oph" to "وف" /of/ or "ف" /f/, but what we may get is "وپه" /oph/ or "په" /ph/ (hardly pronounceable).

We propose new segmentation methods to fulfil the requirements of an effective transliteration system to avoid any blind treatment of characters in the transliteration stage. Consonant-vowel based approaches, in contrast to $v\backslash f$ methods which pre-define the segment lengths, constrain segments based on sequences of consonants or vowels. We therefore introduce extra information, called a *pattern* (to be explained in the following sections), to our transformation rules: $(\hat{S}, \text{pattern}) \rightarrow (\hat{T}, p)$. In this chapter, three novel approaches based on consonant-vowel segmentation are proposed: CV-MODEL1, CV-MODEL2, and CV-MODEL3. Our experimental results demonstrate improvement in transliteration accuracy, particularly for English to Persian transliteration. Hence, this chapter addresses the main research problem of improving machine transliteration effectiveness for the language-pair English and Persian by proposing new segmentation and alignment algorithms. We particularly focus on *what language specific features assist English-Persian and Persian-English transliteration?*

4.2 Consonants and Vowels: Clues for Word Segmentation

All words in a language consist of one or more syllables, which are speech sounds units of a word. Syllables in turn are made of consonants (C) and vowels (V). While vowels can occur on their own, consonants need to occur in a context of vowels [Crystal, 2006]. Each language possess its own range of syllable types; for example, typical sequences in English are CV,

	Vowel	Semi-vowel	Consonant
English	a, e, i, o, u	v, w, y	b, c, d, f, g, h, j, k, l, m, n, p, q, r, s, t
Persian	ا [ɒ],[ɒː],[æ],[ɔ],[o],[e], و [v],[u], ی [j],[i]	ه [h]	ب [b], پ [p], ت [t], ث [s], ج [dʒ], چ [tʃ], ح [h], خ [x], د [d], ذ [z], ر [ɾ], ز [z], ژ [ʒ], س [s], ش [ʃ], ص [s], ض [z], ط [t], ظ [z], ع [ʔ], غ [ɣ], ف [f], ق [ɣ], ک [k], گ [g], ل [l], م [m], ن [n]

Table 4.1: Categorisation of consonants and vowels in English and Persian for consonant-vowel based approaches.

CVC, $CCVC$, and $CVCC$, whereas, typical sequences in Persian are V, VC, CV, CVC, and $CVCC$. Inspired by such a syllabification model, we segment words based on our defined patterns which mainly match with these two sets of standard English and Persian patterns, yet with a non-phonological definition of consonant and vowel (the way their sound functions within a given language word context). We categorise characters independent from their context, but based on their general property of being a consonant or vowel in the language alphabet. We are therefore using a grapheme — as opposed to phoneme — approach to the segmentation problem; consonants and vowels are hereby defined as shown in Table 4.1. To simplify, semi-vowels and approximants are treated according to their target language counterparts. That is, if the equivalent target character is a vowel, the source semi-vowel is also categorised as vowel, and if the target character is consonant, the source is counted as consonant too.

General patterns of transliteration from English to Persian that we analysed in a bilingual corpus, led us to define similar sequences that resemble those of the standard sequences for both languages. For example, we noticed a habit of transliterating "e" in between two consonants such as "hen" to null (ε) in Persian. That is, in a context of CVC where V="e", transliteration of the vowel is ε. Therefore, CVC makes a valid pattern in our transliteration task. Such patterns appear in our models, and are carefully chosen in contexts that mimic the habits of human transliterators. Details are explained in the following sections where we

show how application of these patterns leads to sensible segmentation approaches that avoid any predefined segment length. Before moving to the details of each method separately, let us define the principle terms used to explain the new approaches.

Definition 6 *Given some word, w, the consonant-vowel sequence $r = (C|V)^+$ for w is obtained by replacing each consonant with C and each vowel with V.*

Definition 7 *Given some consonant-vowel sequence, r, a reduced consonant-vowel sequence q replaces all runs of Cs with $\mathcal{C}$, and all runs of Vs with $\mathcal{V}$; hence $q = q'|q''$, $q' = \mathcal{V}(\mathcal{CV})^*(\mathcal{C}|\epsilon)$ and $q'' = \mathcal{C}(\mathcal{VC})^*(\mathcal{V}|\epsilon)$.*

We use the concepts of *collapsed vowels* denoted by $\mathcal{V}$, and *collapsed consonants* denoted by $\mathcal{C}$, as demonstrated in Definitions 6 and 7, to describe our methods. These approaches are differentiated by their segment patterns, and their segment-to-rule matching process.

4.3 Collapsed-Vowel Approach: CV-MODEL1

CV-MODEL1 is a transliteration approach using collapsed vowels and exact matching of segments. The first step in the training phase is the alignment of each word-pair (S, T); then, the consonant-vowel sequence of the source word is formed. This sequence is then parsed based on its consonants and vowels in two ways:

1. C, a segment consisting of a single consonant; and
2. $\mathcal{V}$, a run of consecutive vowels.

After this initial parsing is complete, words are re-segmented using consonant and vowel boundaries to define new segments that we add to the existing ones (a set of already recognised segments). Four groups of segments are defined:

1. $\mathcal{V}C$, a run of vowels at the beginning of a word followed by a consonant;
2. CC, two consonants not separated by any vowels;
3. $C\mathcal{V}C$, a run of vowels bounded by two consonants; and

[1]Phonetic of this letter is various. Just one of them is listed here.

4. $C\mathcal{V}$, a run of vowels at the end of a word preceded by a consonant.

The first consonant in each segment overlaps the final consonant in the preceding segment. In each case the full segment is added to the source segments set (consonants coming after vowels are not transliterated in the preceding matching). For example, consider the word "baird" and its aligned target word:

English:	b	ai	r	d
Persian (left-to-right):	ب	ی	ر	د

This would first be parsed to form the consonant-vowel sequence $CVVCC$ and then initial segmentation resulting in "b", "r" and "d" being added to C segments and "ai" being added to $\mathcal{V}$ segments. Note that vowels are treated differently, considering all of them as one possible segment. In the second phase, the word would be parsed into segments as:

English:	b	ai	r		d
Persian (left-to-right):	ب	ی	ر		د
Segmentation:	C	$\mathcal{V}$	C		C
	Segment 1				Segment 3
			Segment 2		

resulting in possible contexts "bair" and "rd" as $C\mathcal{V}C$, CC, and C segments respectively.

Transformation rule generation is performed similarly to $v\backslash f$ methods. Each segment has a pattern that specifies the translatable part of each segment. For every category of segments, we define translatable symbols: C, V, and CV segments are transliterated completely, and CC, CVC and VC are transliterated without the tailing consonant. For the "baird" example we generate the following rules:

(–, ai ,–,$\mathcal{V}$) → ی,
(–, b ,–,C) → ب,
(–, r ,–,C) → ر,
(–, d ,–,C) → د,
(–, r ,d ,CC) → ر,
(–, bai , r ,$C\mathcal{V}C$) → بی,

where head of each rule follows ($\hat{S}$, pattern) notation where $\hat{S} = c_v, \hat{s}, c_f$, with c_v being past context, c_f future context, and $\hat{s}$ a substring that is transliterated to $\hat{T}$ (probabilities are not shown). The patterns (C, V, $C\mathcal{V}$, CC, $C\mathcal{V}C$, and $\mathcal{V}C$) and their role in defining the $\hat{s}$ substring of each segment can formally be represented as (–,c,–, C), (–,$v_1 \dots v_m$,–, V) (c_1,$v_1 \dots v_m$,–, CV), (–,c_1,c_2, CC), (c_1,$v_1 \dots v_m$,c_2, CVC), and (–,$v_1 \dots v_m$,c_2, VC), where c_1 is the context related to the first C, c_2 is the context related to the second, and $v_1 \dots v_m$ is related to $\mathcal{V}$.

An overview of the training steps is shown in Algorithm 3. The probability assigned to each transformation rule, similar to Equation 3.1 on page 70, is based on the frequencies of the symbols and segments in the training corpus. It can be noted that CV-MODEL1 works based on exact matching of segments in each source word. Therefore, in the segment matching phase, not only we are aware of the role of characters in terms of consonant or vowel in their source language (pattern), but also we value their original alphabetical symbol ($\hat{S}$).

In the transliteration generation stage, segmentation continues using $\mathcal{V}C$, CC, $C\mathcal{V}C$, and $C\mathcal{V}$ segment patterns. If they are not found in the existing segments of these patterns, then they are broken into C and $\mathcal{V}$ segments applying a backoff method. The backoff approach for each of the four patterns is given by the following three rules, which are applied in order:

1. if the context is a $C\mathcal{V}$ segment, and does not occur in the training data, separate the symbol into a C then $\mathcal{V}$ context; else

2. if the context is a $\mathcal{V}$ segment representing a run of more than one vowel, and does not occur in the training data, reduce the length of the run by one and transliterate the final vowel as a single character with no context (for example if search for “ai” was unsuccessful, it looks for “a” and “i” separately); else

3. for all other contexts, if it does not occur in the training data, drop the trailing C and try this shorter context.

These rules are repeatedly applied until a match is found.

Transliteration generation, similar to $v\backslash f$ methods, apply Algorithm 2 in Chapter 3 to generate the ranked list of transliteration suggestions for each test word. A complete example of how the segmentation is performed in the transliteration stage of CV-MODEL1 method, is shown in Figure 4.2.

i	segment	pattern	CV-MODEL1	CV-MODEL2
1	e n	VC	(–,e,n, VC)	(–,e,C, VC)
2	n r	CC	(–,n,r, CC)	(–,n,C, CC)
3	r i q	CVC	(–,r,–, C) and (r,i,q, CVC)	(–,r,–, C) and (C,i,C, CVC)
4	q u e	CV	(–,q,–, C) and (q,ue,–, CV)	(–,q,–, C) and (C,ue,–, CV)

Figure 4.2: Example of the segments generated for the source word "enrique". Columns four and five show the heads of each transformation rule that defines the segments.

4.4 Collapsed-Vowel Approach: CV-MODEL2

In the second technique, CV-MODEL2, we relax the strict matching on the consonant component of the context so that it can match any consonant. Vowels are only used in a context when transliterating a symbol that contains a vowel. This way our knowledge from some contexts reduces to their consonant nature (C) and not the exact alphabetical symbol (c_1 or c_2). Hence, all CVC segments must be split into C and CVC segments. Similarly, CV segments from CV-MODEL1 must be split into C and CV. To explain in terms of transformation rules, they are formed using the following formulation: (C,$v_1 \dots v_m$,–, CV), (–,c_1,C, CC), (C,$v_1 \dots v_m$,C, CVC), and (–,$v_1 \dots v_m$,C, VC).

Repeating the "baird" example given for CV-MODEL1 approach, we define the following rules resulting from segmentation in the training step:

(–, ai ,–, $\mathcal{V}$) → ی,
(–, b ,–, C) → ب,
(–, r ,–, C) → ر,
(–, d ,–, C) → د,
(–, r ,–, CC) → ر,
(–, ai ,–, $C\mathcal{V}C$) → ی.

Note the acceptable patterns are the same as CV-MODEL1: C, $\mathcal{V}$, CC, $C\mathcal{V}C$, $C\mathcal{V}$, and $\mathcal{V}C$. In the example above, in contrary to CV-MODEL1, for $C\mathcal{V}C$ pattern we only keep ai part of "bair" segment letting any $CaiC$ context be matched with this transformation rule.

Algorithm 3 Consonant-vowel transliteration: training stage.

Require: Acceptable patterns of the transliteration approach (e.g. C, $\mathcal{V}$, CC, $C\mathcal{V}C$, $C\mathcal{V}$, and $\mathcal{V}C$ for CV-MODEL1).

1: Let $B' = \{(S, T)\}$ be the training corpus of source and target word pairs.
2: **for** each $(S, T) \in B'$ **do**
3: Align (S, T) substrings.
4: Generate consonant-vowel sequence r of the source word S.
5: Segment S based on the acceptable patterns found in r.
6: Make a rule $(\hat{S}, \text{pattern}) \rightarrow \hat{T}$ for each segment $\hat{S}$ of S.
7: **end for**
8: **for** each transformation rule generated in the previous step **do**
9: $p = \frac{\text{frequency of } \hat{S} \rightarrow \hat{T}}{\text{frequency of } \hat{S}}$.
10: Add the rule $(\hat{S}, \text{pattern}) \rightarrow (\hat{T}, p)$ to transliteration model.
11: **end for**
12: Output the transliteration model.

Thus, if for example the word "Rain" is in test collection, in order to transliterate vowels, the segment "rain" matches to a rule categorised under CVC pattern; thus increasing the amount of training data for a context. In turn, this leads to rare, but existent, transliterations that would otherwise not be available.

A transliteration generation example is shown in the final column of Figure 4.2 for the string "enrique". The backoff strategy is similar to the CV-MODEL1 method.

4.5 Collapsed Consonant-Vowel Approach: CV-MODEL3

The restriction on the context length of consonants imposed by CV-MODEL1 and CV-MODEL2 makes the transliteration of consecutive consonants mapping to a particular character in the target language difficult. For example, "ght" in English maps to only one character in Persian: "ت" [t]. Dealing with languages which have different alphabets, and for which the number of characters in their alphabets also differs (such as 26 and 32 for English and Persian), increases the possibility of facing these cases, especially when moving from the language with smaller alphabet size to the one with a larger size. To more effectively address

i	segment	pattern	CV-MODEL3	backoff
1	# s c h	$\mathcal{C}$	(–,sch,–, $\mathcal{C}$)	(–,s,–, $\mathcal{C}$) and (–,ch,–, $\mathcal{C}$)
2	h a f	$\mathcal{CVC}$	(h,a,f, $\mathcal{CVC}$)	(–,a,–, $\mathcal{CVC}$) or (–,a,–, $\mathcal{V}$)
3	f f	$\mathcal{C}$	(–,ff,–, $\mathcal{C}$)	(–,f,–, $\mathcal{C}$)
4	f e r	$\mathcal{CVC}$	(f,e,r, $\mathcal{CVC}$)	(–,e,–, $\mathcal{CVC}$) or (–,e,–, $\mathcal{V}$)
5	r #	$\mathcal{C}$	(–,r,–, $\mathcal{C}$)	ε

Figure 4.3: Example of the segments, patterns, and matching transformation rule heads, for the test word "schaffer" using CV-MODEL3.

this, we propose a *collapsed consonant and vowel* method (CV-MODEL3) which uses the full reduced sequence (Definition 7), rather than just collapsed vowel sequences. Therefore, in CV-MODEL3 we do not have any CC pattern, having it generalised to a $\mathcal{C}$ pattern to consist any number of sequential consonants. Although recognition of consonant segments is still based on vowel positions in the very first parsing, consonants are considered as independent blocks of each string, where vowels are transliterated in the context of consonants around.

Similar to $v\backslash f$ methods, we consider a special symbol to indicate the start and end of each word in case it is not known based on pattern ($\mathcal{VC}$ and $\mathcal{CV}$). Therefore, for words starting or ending with consonants, the symbol "#" has been added which is treated as a consonant and, therefore, grouped in the consonant segment. An example of applying this technique is shown in Figure 4.3 for the word "schaffer".

This method also follows the same steps of other consonant-vowel based methods except for segmentation. In the "Schaffer" example, "sch" and "ff" are treated as two consonant-only segments where the transliteration of individual characters inside a cluster is dependant on other members, but not the surrounding segments. That is, context is defined based on the including consonants without any external influence. However, this is not the case for vowels which are more context conscious and have one level of knowledge about surrounding segments.

4.6 Experimental Setup

To evaluate consonant-vowel methods and compare them to n-gram based transliteration approaches discussed previously in Chapter 3, we use the same corpora: B_{e2p} and B^{+}_{e2p} for English-Persian transliteration, and B_{p2e} for Persian-English transliteration. As explained in Section 2.5 on page 58, all experiments are performed using 10-fold cross-validation, and the results are averaged over 10 runs. The performance of systems is evaluated using word accuracy and character accuracy metrics, and the difference of their effectiveness is judged based on the paired t-test statistical significance test.

4.7 Comparison of Segmentation Algorithms

A comparison of effectiveness of consonant-vowel methods and baseline methods is presented in this section. The most effective transliteration systems from the previous chapter, n-gram based approaches, are chosen as baseline; English to Persian transliteration was best using 1\0, one past context, and Persian to English transliteration performed most effectively with 2\2T.

Results of comparisons between the baseline and CV-MODEL1 method are reported in Table 4.2, categorised by corpus. Evaluating on B_{e2p}, CV-MODEL1 is more accurate than 1\0 in its first suggesting transliteration (TOP-1), improving mean word accuracy by a relative 6.4%, from 58.0% to 61.7%. When evaluating on B^{+}_{e2p}, the improvement is 9.4% in transliterations of TOP-1 rank, giving a 51.6% translation accuracy. Although evaluations on the larger corpus shows improvement in total of TOP-1 to TOP-20 results for English-Persian transliteration, we conclude that CV-MODEL1 highly ranked transliterations are significantly more accurate than the baseline system 1\0.

Persian to English transliteration also benefits from CV-MODEL1 with improved word accuracy from TOP-1 to TOP-20; with the highest improvement seen in TOP-1 with a significant relative increase of 12.3%, which is an increase from 47% to 52.8%.

Mean character accuracy results are shown in Table 4.3. On average, in terms of character accuracy, only small improvements are gained for the three corpora. English to Persian transliteration is improved 1.1% for the B_{e2p} corpus. Improvement in mean character accuracy for Persian to English transliteration is absolute 1.6%.

The results for the CV-MODEL2 method are shown in Table 4.4. CV-MODEL2 in contrast to the CV-MODEL1 method, which was more accurate on top-ranked transliterations in comparison to the baseline, is more accurate in its TOP-10 to TOP-20 transliterations. English-Persian transliteration on B_{e2p} is significantly improved in TOP-1 by 3.4% relatively (paired t-test, p=0.0521). The relative improvement of 2.5% in TOP-1 on the B^{+}_{e2p} corpus is not statistically significant. Persian-English transliteration accuracy, however, is hurt using the CV-MODEL2 approach, only showing improvements from TOP-10 to TOP-20 results. Character accuracy also does not differentiate between the baseline and CV-MODEL2 methods in TOP-1 English-Persian transliterations (Table 4.5). Persian-English transliteration is better achieved by the baseline system than CV-MODEL2, which decreases the mean character accuracy on TOP-1 transliteration by 2.1%.

CV-MODEL3 transliteration results are given in Table 4.6, comparing mean word accuracy with the baseline methods. According to this table, CV-MODEL3 shows a 16.2% relative improvement over the baseline for B_{e2p}, and a 17.2% relative improvement for B^{+}_{e2p}. The positive impact of the CV-MODEL3 method is consistently extended to both corpora, and all ranks of system output. The results for mean character accuracy confirm the superiority of CV-MODEL3 to the baseline in TOP-1 rank (Table 4.7). CV-MODEL3 improves accuracy by 3.6% and 3.8% for the B_{e2p} and B^{+}_{e2p} corpora, respectively. Persian to English transliteration, in contrast, does not benefit from this method, showing a decrease of 8.0% mean word accuracy in TOP-1 results. Character accuracy is also lower when using CV-MODEL3, showing a 2.1% decrease.

The number of segments generated for all the consonant-vowel based transliteration methods is reported in Table 4.8. CV-MODEL1 method which saves all the source word segments in their original shape, has the largest number of source, target, and source to target rules segments.

To summarise, experiments suggest that CV-MODEL3 which uses collapsed consonants and vowels is most beneficial for English to Persian transliteration, significantly improving the top-ranked transliteration by 17.2% for the B^{+}_{e2p} corpus which was poorly transliterated by most n-gram based approaches. Persian to English transliteration is improved using CV-MODEL1, showing an increase of 12.3%. This difference is caused by the lack of vowels in written Persian. Most source words that fail to be segmented are sequences of consonants

Corpus		Baseline	CV-MODEL1	p-value	Absolute Improvement	Relative Improvement
B_{e2p}	TOP-1	58.0 (3.4)	61.7 (3.0)	< 0.0001	+3.7	+6.4
	TOP-5	85.6 (2.2)	80.9 (2.2)	< 0.0001	-4.7	-5.8
	TOP-10	89.4 (2.9)	82.0 (2.1)	< 0.0001	-7.4	-8.3
	TOP-20	90.5 (3.1)	82.2 (2.3)	< 0.0001	-8.3	-9.7
B^{+}_{e2p}	TOP-1	47.2 (1.0)	51.6 (2.5)	0.0007	+4.4	+9.3
	TOP-5	77.6 (1.4)	79.8 (3.4)	< 0.0001	+2.2	+2.8
	TOP-10	83.3 (1.5)	84.9 (3.1)	0.1494	+1.6	+1.9
	TOP-20	86.1 (1.4)	87.0 (3.2)	0.4428	+0.9	+1.0
B_{p2e}	TOP-1	47.0 (2.4)	52.8 (3.2)	0.0016	+5.8	+12.3
	TOP-5	66.5 (3.1)	72.7 (4.6)	< 0.0001	+6.2	+9.3
	TOP-10	70.3 (3.4)	77.1 (3.7)	< 0.0001	+6.8	+8.8
	TOP-20	72.7 (2.8)	79.2 (3.2)	< 0.0001	+6.5	+8.9

Table 4.2: Comparison of mean word accuracy between the baseline system and CV-MODEL1*. Baseline for English-Persian is* $1\backslash 0$*, and for Persian-English is* $2\backslash 2T$*. Standard deviations are given in the parentheses, and all the numbers are percentages.*

	Method		p-value
Corpus	Baseline	CV-MODEL1	(Baseline,CV-MODEL1)
B_{e2p}	89.2 (1.0)	90.3 (1.0)	0.0016
B^{+}_{e2p}	85.5 (0.5)	86.5 (0.7)	0.0626
B_{p2e}	87.0 (1.2)	88.6 (1.1)	0.0054

Table 4.3: Mean character accuracy (TOP-1) *for* CV-MODEL1*. Standard deviations are given in the parentheses, and all the numbers are percentages.*

only (one $\mathcal{C}$ segment is generated).

Corpus		Baseline	CV-MODEL2	p-value	Absolute Improvement	Relative Improvement
B_{e2p}	TOP-1	58.0 (3.4)	60.0 (3.9)	0.0521	+2.0	+3.4
	TOP-5	85.6 (2.2)	86.0 (2.8)	0.6830	+0.4	+0.5
	TOP-10	89.4 (2.9)	91.2 (2.5)	0.0007	+1.8	+2.0
	TOP-20	90.5 (3.1)	93.7 (2.1)	0.0292	+3.2	+3.5
B^{+}_{e2p}	TOP-1	47.2 (1.0)	48.4 (1.0)	0.4528	+1.2	+2.5
	TOP-5	77.6 (1.4)	79.2 (1.0)	0.0054	+1.6	+2.1
	TOP-10	83.3 (1.5)	87.0 (0.9)	< 0.0001	+3.7	+4.4
	TOP-20	86.1 (1.4)	91.8 (0.9)	< 0.0001	+5.7	+6.6
B_{p2e}	TOP-1	47.0 (2.4)	36.5 (2.6)	< 0.0001	-10.5	-22.3
	TOP-5	66.5 (3.1)	64.0 (3.2)	0.0228	-2.5	-3.8
	TOP-10	70.3 (3.4)	73.2 (2.5)	0.0177	+2.9	+4.1
	TOP-20	72.7 (2.8)	78.6 (2.4)	< 0.0001	+5.9	+8.1

Table 4.4: Comparison of mean word accuracy for baseline and CV-MODEL2*. Baseline for English-Persian is* $1\backslash 0$ *and for Persian-English is* $2\backslash 2T$*. Standard deviations are given in the parentheses.*

	Method		p-value
Corpus	Baseline	CV-MODEL2	(Baseline,CV-MODEL2)
B_{e2p}	89.2 (1.0)	89.7 (1.1)	0.0925
B^{+}_{e2p}	85.5 (0.5)	85.5 (0.2)	0.4287
B_{p2e}	87.0 (1.2)	84.9 (0.8)	0.0008

Table 4.5: Mean character accuracy (TOP-1) *for* CV-MODEL2*.*

Corpus		Baseline	CV-MODEL3	p-value	Absolute Improvement	Relative Improvement
B_{e2p}	TOP-1	58.0 (3.4)	67.4 (5.5)	< 0.0001	+9.4	+16.2
	TOP-5	85.6 (2.2)	90.9 (2.1)	< 0.0001	+5.3	+6.2
	TOP-10	89.4 (2.9)	93.8 (2.1)	< 0.0001	+4.4	+4.9
	TOP-20	90.5 (3.1)	94.7 (2.1)	< 0.0020	+4.2	+4.6
B^{+}_{e2p}	TOP-1	47.2 (1.0)	55.3 (0.8)	< 0.0001	+8.1	+17.2
	TOP-5	77.6 (1.4)	84.5 (0.7)	< 0.0001	+6.9	+8.9
	TOP-10	83.3 (1.5)	89.5 (0.4)	< 0.0001	+6.2	+7.4
	TOP-20	86.1 (1.4)	92.4 (0.5)	< 0.0001	+6.3	+7.3
B_{p2e}	TOP-1	47.0 (2.4)	39.0 (2.6)	< 0.0001	-8.0	-17.0
	TOP-5	66.5 (3.1)	68.0 (2.7)	0.1259	+1.5	+2.2
	TOP-10	70.3 (3.4)	75.3 (2.6)	0.0012	+5.0	+7.1
	TOP-20	72.7 (2.8)	79.1 (2.2)	< 0.0001	+6.4	+8.8

Table 4.6: Comparison of mean word accuracy between the baseline system and CV-MODEL3*. Baseline for English-Persian is* $1\backslash 0$ *and for Persian-English is* $2\backslash 2T$*. Standard deviations are given in the parentheses.*

4.8 English to Persian Character Alignment

Previous work on transliteration either employs a word alignment tool (usually GIZA++), or develops specific alignment strategies. Transliteration methods that use GIZA++ as their word pair aligner [AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003b] have based their work on the assumption that the provided alignments are reliable. Gao et al. [2004a;b] argue that precise alignment can improve transliteration effectiveness, experimenting on English-Chinese data and comparing the IBM models [Brown et al., 1993] with phoneme-based alignments using direct probabilities.

Other transliteration systems focus on alignment for transliteration (monotonous alignment versus non-monotonous alignment suitable for translation), for example the joint source-channel model suggested by Li et al. [2004]. Their method outperforms the noisy channel

	Method		p-value
Corpus	Baseline	CV-MODEL3	(Baseline,CV-MODEL3)
B_{e2p}	89.2 (1.0)	92.8 (1.4)	< 0.0001
B^{+}_{e2p}	85.5 (0.5)	89.3 (0.2)	< 0.0001
B_{p2e}	87.0 (1.2)	84.9 (0.8)	< 0.0001

Table 4.7: Mean character accuracy (TOP-1) *for* CV-MODEL3.

		Method		
	Corpus	CV-MODEL1	CV-MODEL2	CV-MODEL3
Source	B_{e2p}	1,129	286	485
	B^{+}_{e2p}	3,768	593	1,472
	B_{p2e}	1,148	414	1,074
Target	B_{e2p}	1,129	286	485
	B^{+}_{e2p}	3,768	593	1,472
	B_{p2e}	2,258	884	1,128
Source-Target	B_{e2p}	1,324	371	69
	B^{+}_{e2p}	6,945	1,508	575
	B_{p2e}	2,429	1,202	1,864

Table 4.8: Average number of segments generated in each consonant-vowel method.

model in direct orthographical mapping for English-Chinese transliteration. Li et al. also find that grapheme-based methods that use the joint source-channel model are more effective than phoneme-based methods due to removing the intermediate phonetic transformation step. Alignment has also been investigated for transliteration by adopting Covington's algorithm on cognate identification [Covington, 1996]; this is a character alignment algorithm based on matching or skipping of characters, with a manually assigned cost of association. Covington considers consonant to consonant and vowel to vowel correspondence more valid than consonant to vowel. Kang and Choi [2000] revise this method for transliteration where a skip is defined as inserting a null in the target string when two characters do not match

based on their phonetic similarities or their consonant and vowel nature. Oh and Choi [2002] revise this method by introducing *binding*, in which many to many correspondences are allowed. However, all of these approaches rely on manually assigned penalties that need to be defined for each possible matching.

The transliteration experiments we reported so far, using n-gram based and consonant-vowel based methods, all use GIZA++ in their training stage. Although consonant-vowel methods partially remove the dependency of transliteration rules on alignment, they can still be affected by misalignments. To solve this problem, two research questions should be answered:

1. how does the accuracy of alignments affect the accuracy of transliteration; and,

2. can we improve transliteration accuracy by improving the alignment process?

We therefore investigate the effect of an alternative approach to GIZA++ on accuracy. Our method is limited to English to Persian alignment, however; due to lack of written short vowels in Persian script we cannot apply this method successfully for the Persian to English alignment task.

We propose an alignment method based on segment occurrence frequencies, thereby avoiding predefined matching patterns and penalty assignments. We also apply the observed tendency of aligning consonants to consonants, and vowels to vowels, as a substitute for phonetic similarities. Many to many, one to many, one to null, and many to one alignments can be generated.

Our alignment approach consists of two steps: the first is based on the consonant and vowel nature of the word's letters, while the second uses a frequency-based sequential search.

For each natural language word, we can determine the consonant-vowel sequence r (Definition 6 on page 97) from which the *reduced consonant-vowel* sequence q (Definition 7) can be derived, giving a common notation between two different languages, no matter which script either of them use.

In general, for all the word pairs (S, T) in a corpus B, an alignment can be achieved using the function

$$f : B \to \mathcal{A}; (S, T) \mapsto (\hat{S}, \hat{T}, n).$$

The function f maps the word pair $(S, T) \in B$ to the triple $(\hat{S}, \hat{T}, n) \in \mathcal{A}$ where $\hat{S}$ and

$\hat{T}$ are substrings of S and T respectively. The frequency of this correspondence is denoted by n, representing its occurrence in the whole training corpus. While $\mathcal{A}$ represents a current set of substring alignments (used during the training stage), we use a notation of a_{e2p} for each specific transliterated word-pair (S, T) when aligning English to Persian and a_{p2e} for back-transliteration Persian out-of-dictionaries to English. The alignments registered as a_{e2p} and a_{p2e} form the outputs of the alignment task.

4.8.1 Alignment Algorithm

Our algorithm consists of two steps.

Step 1: Consonant-Vowel based

For any word pair $(S, T) \in B$, the corresponding *reduced consonant-vowel* sequences, q_S and q_T, are generated. If the sequences match, then the aligned consonant clusters and vowel sequences are added to the alignment set $\mathcal{A}$ updating n values gradually, and keeping the aligned pair (a_{e2p}) for output. If q_S does not match with q_T, the word pair remains unaligned in *Step 1*.

The assumption in this step is that transliteration of each vowel sequence of the source is a vowel sequence in the target language, and similarly for consonants. However, consonants do not always map to consonants, or vowels to vowels (for example, the English letter "s" may be written as "اس" /es/ in Persian which consists of one vowel and one consonant). Alternatively, they might be omitted altogether, which can be specified as the null string, ε. We therefore require a second step.

Step 2: Frequency based

For most natural languages, the maximum length of corresponding phonemes of each grapheme is a digraph (two letters) or at most a trigraph. Hence, alignment can be defined as a search problem that seeks for units with a maximum length of two or three in both strings that need to be aligned. In our approach, we search based on statistical occurrence of data available from *Step 1*.

In *Step 2*, only those words that remain unaligned at the end of *Step 1* need to be considered. For each pair of words (S, T), matching proceeds from left to right, examining

one of the three possible options of transliteration: single letter to single letter, digraph to single letter and single letter to digraph. Trigraphs are unnecessary in alignment as they can be effectively captured during transliteration generation, as we explain below.

We define four different valid alignments for the source ($S = s_1 s_2 \dots s_i \dots s_l$) and target ($T = t_1 t_2 \dots t_j \dots t_m$) strings: (s_i, t_j, n), $(s_i s_{i+1}, t_j, n)$, $(s_i, t_j t_{j+1}, n)$ and (s_i, ε, n). These four options are considered as the only possible valid alignments, and the most frequently occurring alignment (highest n) is chosen. These frequencies are dynamically updated immediately after successfully aligning a pair. For exceptional situations, where there is no character in the target string to match with the source character s_i, it is aligned with the empty string.

It is possible that none of the four valid alignment options have occurred previously (that is, $n = 0$ for each). This situation can arise in two ways: first, such a tuple may simply not have occurred in the training data; and, second, the alignment to the left of the current position may have been incorrect. To account for this second possibility, a partial backtracking is considered. Most misalignments are derived from the simultaneous comparison of alignment possibilities, giving the highest priority to the most frequent. For example if S=bbc, T=س ب /bs/ and $\mathcal{A}$ = {(b,ب,100),(bb,ب,40),(c,س,60)}, starting from the initial position s_1 and t_1, the first alignment choice is (b,ب,100). However, immediately after we face the problem of aligning the second "b" with the Persian "س" [s]. There are two solutions: inserting ε and adding the triple (b,ε,1), or backtracking the previous alignment and substituting that with the less frequent but possible alignment of (bb,ب,40). The second solution is a better choice as it adds less ambiguous alignments containing ε. At the end, the alignment set is updated as $\mathcal{A}$ = {(b,ب,100),(bb,ب,41),(c,س,61)}.

In case of equal frequencies, we check possible subsequent alignments to decide on which alignment should be chosen. For example, if (b,ب,100) and (bb,ب,100) both exist as possible options, we consider if choosing the former leads to a subsequent ε insertion. If so, we opt for the latter. In other words, if (b,ب,100) is chosen, then the second "b" has no match in the target string which means it aligns to ε.

At the end of a string, if just one character in the target string remains unaligned while the last alignment is a ε insertion, that final alignment will be substituted for ε. This usually happens when the alignment of final characters is not yet registered in the alignment

Initial alignment set:	
$\mathcal{A}$ = {(p,پ,42),(a,ا,320),(a,ε,99),(a,ا‌ی,10),(a,ی,35),(r,ر,200),(i,ی,60),(i,ε,5),(c,س,80),(c,چ,25),(t,ت,51)}	
Input:	(patricia,ای‌ش‌ی‌رت‌پ) $q_S = \mathcal{CVCVCV}$ $q_T = \mathcal{CVCV}$
Step 1:	$q_S \neq q_T$
Forward alignment:	(p,پ,43), (a,ε,100), (t,ت,52), (r,ر,201), (i,ی,61), (c,ε,1), (i,ε,6), (a,ε,100)
Backward alignment:	(a,ا,321), (i,ی,61), (c,ε,1), (i,ε,6), (r,ε,1), (t,ε,1), (a,ε,100), (p,ε,1)
Input 2:	(ici,ی‌ش‌ی) $q_S = \mathcal{VCV}$ $q_T = \mathcal{VCV}$
Step 1:	(i,ی,61),(c,ش,1), (i,ی,61)
Final Alignment:	a_{e2p} = ((p,پ),(a,ε),(t,ت),((r,ر),(i,ی),(c,ش),(i,ی),(a,ا))
Updated alignment set:	
$\mathcal{A}$ = {(p,پ,43),(a,ا,321),(a,ε,100),(a,ا‌ی,10),(a,ی,35),(r,ر,201),(i,ی,62),(i,ε,5),(c,س,80),(c,چ,25), (c,ش,1),(t,ت,52)}	

Figure 4.4: A back-parsing example. Note middle tuples in forward and backward parsings are not merged in $\mathcal{A}$ until the alignment is successfully completed.

set, mainly because Persian speakers tend to transliterate the final vowels to consonants to preserve their existence in the word. For example, in the word "Jose" the final "e" might be transliterated to "ه" [h] which is a consonant and therefore is not captured in *Step 1*.

Back-parsing

The process of aligning words explained above can handle words with already known components in the alignment set $\mathcal{A}$ (the frequency of occurrence is greater than zero). However, when this is not the case, the system may repeatedly insert ε while part or all of the target characters are left intact (unsuccessful alignment). In such cases, processing the source and target backwards helps to find the problematic substrings: *backparsing*.

Substrings that are poorly aligned both forwards and backwards in the source and target are taken as new pairs of strings, which are then reintroduced into the system as new entries. Note that they themselves are not subject to back-parsing. Most strings of repeating nulls can be broken up this way, and in the worst case will remain as one tuple in the alignment set.

To clarify, consider the example given in Figure 4.4. For the word pair (patricia,ای‌ش‌ی‌رت‌پ), where an association between "c" and "ش" [ʃ] is not yet registered. Forward parsing, as shown

in the figure, does not resolve all target characters; after the incorrect alignment of "c" with "ε", subsequent characters are also aligned with null, and the substring "ای‌ش" /ʃiːɒ/ remains intact. Backward parsing, shown in the next line of the figure, is also not successful. It is able to correctly align the last two characters of the string, before generating repeated null alignments. Therefore, the central region — substrings of the source and target which remained unaligned plus one extra aligned segment to the left and right — is entered as a new pair to the system (ici,ی‌شی), as shown in the line labelled *Input 2* in the figure. This new input meets *Step 1* requirements, and is aligned successfully. The resulting tuples are then merged with the alignment set $\mathcal{A}$.

An advantage of our back-parsing strategy is that it takes care of casual transliterations happening due to elision and epenthesis. It is not only in translation that people may add extra words to make fluent target text; for transliteration also, it is possible that spurious characters are introduced for fluency. However, this often follows patterns, such as adding vowels to the target form. These irregularities are consistently covered in the back-parsing strategy, where they remain connected to their previous character.

All the explained steps of our alignment method are summarised in Algorithm 4.

4.8.2 Impact of Alignment on Transliteration Accuracy

The results of applying our new alignment algorithm in the transliteration process are presented in Table 4.9, comparing word accuracy of CV-MODEL3 using GIZA++ and the new alignment for English to Persian transliteration. Transliteration accuracy increases in TOP-1 for both corpora (a relative increase of 7.1% (p=0.0155) for the B_{e2p} corpus and 8.1% (p=0.0002) for the B^{+}_{e2p} corpus). The TOP-10 results of the B^{+}_{e2p} again show a relative increase of 3.5% (p=0.0004). Although the new alignment also increases the performance for TOP-5 and TOP-20 of the B^{+}_{e2p} corpus, these increases are not statistically significant.

4.9 English to Persian Back-Transliteration

Transliteration is a productive process; for one word there may be many possible acceptable transliterations. Back-transliteration, however, is more strict and expects to re-generate the exact original source word. Considering the information already lost in forward transliteration, back-transliteration is a challenging task. For instance, transliteration of all the three

Algorithm 4 Substring-level alignment of source and target pairs.

1: Let $B' = \{(S, T)\}$ be the training corpus of source and target word pairs.
2: Let $\mathcal{A} \leftarrow \emptyset$ represent the alignment set with ($\hat{S}$,$\hat{T}$, frequency) elements.
3: **while** there is an unprocessed (S, T) in B' **do**
4: Form q_S and q_T as reduced consonant-vowel sequences of S and T, respectively.
5: **if** $q_S = q_T$ **then**
6: Align substrings of consonants ($\mathcal{C}$) together and vowels ($\mathcal{V}$) together.
7: Add the newly found elements, and update the frequency of the aligned substrings in $\mathcal{A}$.
8: Record the aligned pair.
9: **else**
10: Record (S, T) in a set of unaligned pairs B''.
11: **end if**
12: **end while**
13: **while** there is an unprocessed (S, T) in B'' **do**
14: **while** (S, T) is not aligned **do**
15: **repeat**
16: Align (S, T) by forward parsing with the aid of the alignments registered in $\mathcal{A}$.
17: **until** S and T are completely aligned or one of them is aligned while the other still needs processing.
18: **if** alignment is incomplete **then**
19: Do a back-parse to fix the successive null-aligned substrings.
20: **end if**
21: **end while**
22: Add the newly found elements, and update the frequency of the aligned substrings in $\mathcal{A}$.
23: Record the aligned pair.
24: **end while**
25: Output the aligned corpus.

Corpus		GIZA++	New Alignment	p-value	Absolute Improvement	Relative Improvement
B_{e2p}	TOP-1	67.4 (5.5)	72.2 (2.2)	0.0155	+4.8	+7.1
	TOP-5	90.9 (2.1)	92.9 (1.6)	0.0005	+2.0	+2.2
	TOP-10	93.8 (2.1)	93.5 (1.7)	0.1023	-0.5	-0.5
	TOP-20	94.7 (2.1)	93.6 (1.8)	0.0615	-1.1	-1.2
B^{+}_{e2p}	TOP-1	55.3 (0.8)	59.8 (1.4)	0.0002	+4.5	+8.1
	TOP-5	84.5 (0.7)	85.4 (0.8)	0.5302	+0.9	+1.1
	TOP-10	89.5 (0.4)	92.6 (0.7)	0.0004	+3.1	+3.5
	TOP-20	92.4 (0.5)	92.8 (0.4)	0.8471	+0.4	+0.4

Table 4.9: Comparison of mean word accuracy between the CV-MODEL3 *that uses* GIZA++ *in its alignment and* CV-MODEL3 *using the new alignment for English-Persian transliteration.*

English strings “bus”, “boss”, and “bass” is “باس” /bɒs/. Back-transliterating “باس” is therefore ambiguous. Another problem is when, from multiple transliterations, the system needs to generate only one variant which is acceptable in the original language. For example, “تام” /tɒm/ and “تم” /tom/ are two acceptable transliterations of “Tom”. A back-transliterator should transliterate both to exactly the same word. In this section, we introduce a method of back-transliteration of already transliterated Persian words to English addressing two of the research questions:

1. how does different transliteration methods examined for forward transliteration perform in the backward transliteration task; and,

2. how can these methods be modified to create a back-transliteration system which generates English words from their transliterated Persian word?

Written Persian ignores short vowels, and only long vowels appear in text. This causes most English vowels to disappear when transliterating from English to Persian; hence, these vowels must be restored during back-transliteration.

When the initial transliteration happens from English to Persian, the transliterator (whether human or machine) uses the rules of transliterating from English as the source

language. Therefore, transliterating back to the original language should consider the original process, to avoid information loss. In terms of segmentation in consonant-vowel methods, different patterns define segment boundaries, in which vowels are necessary clues. Although we do not have most of these vowels in the transliteration generation phase, it is possible to benefit from their existence in the training phase. For example, using CV-MODEL3, the pair (لکرم,merkel) with q_S=$\mathcal{C}$ and a_{p2e}=((م,me),(ر,r),(ک,ke),(ل,l)), produces just one transformation rule "لکرم → merkel" based on a $\mathcal{C}$ pattern. That is, the Persian string contains no vowel characters. If, during the transliteration generation phase, a source word "مرکل" /merkel/ is entered, there would be one and only one output of "merkel", while an alternative such as "mercle" might be required instead. To avoid over-fitting the system by long consonant clusters, we perform segmentation based on the English q sequence, but categorise the rules based on their Persian segment counterparts. That is, for the pair (لکرم,merkel) with a_{e2p}=((m,م),(e,ε),(r,ر),(k,ک),(e,ε),(l,ل)), these rules are generated (with category patterns given in parenthesis): م → m ($\mathcal{C}$), رک → rk ($\mathcal{C}$), ل → l ($\mathcal{C}$), رکم → merk ($\mathcal{C}$), رکل → rkel ($\mathcal{C}$). We call the suggested training approach *reverse segmentation*.

Reverse segmentation avoids clustering all the consonants in one rule, since many English words might be transliterated to all-consonant Persian words.

Accuracy of Back-Transliteration

In this section we evaluate back-transliteration using the previously examined transliteration methods, plus our new back-transliteration approach. We examined three n-gram based methods that apply no context and past context (0\0, 1\0, and 2\0) to back-transliterate B_{e2p} and B^{+}_{e2p} Persian words to English. The results are shown in Table 4.10 indicating that methods used for forward transliteration are not as effective at back-transliteration. Comparing 1\0, the best n-gram based method for English-Persian transliteration, in both Tables 4.10 and 3.1 on page 81 in Chapter 3, we experience a drop of 34.9% word accuracy in TOP-1 results evaluating on the B_{e2p} corpus. Consonant-vowel methods also fall in accuracy, as reported in Table 4.11. Directly applying CV-MODEL3 method for B_{e2p} (TOP-1) shows a 47.2% and 31.3% drop in comparison to its forward counterpart for B_{e2p} and B^{+}_{e2p}, respectively.

Comparing the consonant-vowel methods together, Table 4.11, CV-MODEL3 using GIZA++

		Method		
Corpus		0\0	1\0	2\0
B_{e2p}	TOP-1	18.1 (2.4)	25.0 (4.4)	18.8 (4.0)
	TOP-5	34.3 (4.7)	45.1 (4.6)	32.2 (4.1)
	TOP-10	43.3 (4.2)	53.5 (4.2)	37.6 (3.8)
	TOP-20	51.5 (3.8)	59.5 (3.6)	42.8 (4.1)
B^{+}_{e2p}	TOP-1	8.8 (0.5)	9.2 (0.5)	6.6 (0.9)
	TOP-5	16.6 (0.6)	17.6 (0.5)	15.8 (0.6)
	TOP-10	22.0 (0.7)	23.5 (0.6)	21.5 (0.6)
	TOP-20	29.7 (0.6)	31.1 (0.8)	28.3 (0.7)

Table 4.10: Mean word accuracy (%) using a past context for back-transliteration. Standard deviations are given in parentheses.

outperforms 1\0, CV-MODEL1 and CV-MODEL2 by 12.8% to 40.7% (p<0.0001) in TOP-1 for the B_{e2p} corpus. The corresponding improvement for the B^{+}_{e2p} corpus is 12.8% to 74.2% (p<0.0001).

The fourth column of the table shows the performance increase when using CV-MODEL3 with the new alignment algorithm: for the B^{+}_{e2p} corpus, the new alignment approach gives a relative increase in accuracy of 15.5% for TOP-5 (p<0.0001) and 10% for TOP-10 (p=0.005). The new alignment method does not show a significant difference using CV-MODEL3 for the B_{e2p} corpus.

The final column of Table 4.11 shows the performance of the CV-MODEL3 with the new reverse segmentation approach. Reverse segmentation leads to a significant improvement over the new alignment approach in TOP-1 results for B_{e2p} by 40.1% (p<0.0001), and 49.4% (p<0.0001) for the B^{+}_{e2p} corpus.

Corpus		CV-MODEL1	CV-MODEL2	CV-MODEL3		
				GIZA++	New Alignment	Reverse
B_{e2p}	TOP-1	28.8 (4.6)	24.9 (2.8)	32.5 (3.6)	34.4 (3.8)	48.2 (2.9)
	TOP-5	51.0 (4.8)	52.9 (3.4)	56.0 (3.5)	54.8 (3.7)	68.1 (4.9)
	TOP-10	58.2 (5.3)	63.2 (3.1)	64.2 (3.2)	63.8 (3.6)	75.7 (4.2)
	TOP-20	62.1 (4.9)	70.0 (3.5)	69.9 (3.2)	72.0 (3.7)	76.0 (3.9)
B^{+}_{e2p}	TOP-1	15.6 (1.0)	12.0 (1.0)	17.6 (0.8)	18.0 (1.2)	26.9 (0.7)
	TOP-5	31.7 (0.9)	28.0 (0.7)	36.2 (0.5)	41.8 (1.2)	41.3 (1.7)
	TOP-10	40.1 (1.1)	37.4 (0.8)	46.0 (0.8)	50.6 (1.1)	49.3 (1.6)
	TOP-20	48.0 (1.1)	46.5 (0.8)	54.8 (0.7)	57.0 (1.8)	51.2 (1.5)

Table 4.11: Comparison of mean word accuracy for back-transliteration using consonant-vowel methods. Standard deviations are given in parentheses.

4.10 Summary

Novel approaches for transliteration are presented in this chapter and compared with the n-gram based methods as baseline. Our attempt to include language information in the transliteration process led us to propose consonant-vowel based approaches that are more creative in their segmentation step using consonant and vowel boundaries of each word. We explored the effect of consonant-vowel based methods, CV-MODEL1, CV-MODEL2 and CV-MODEL3, on transliteration effectiveness. Taken together, consonant-vowel approaches provide higher accuracy than n-gram based methods for both English to Persian, and Persian to English transliteration, answering the research question: what language specific features assist English-Persian and Persian-English transliteration?

Transliterating English to Persian the three novel methods, showed significant improvement compared to the 1\0 method as baseline. According to the results, CV-MODEL3 is the best English-Persian method with highest word and character accuracy in TOP-1 to TOP-20, evaluated on both B_{e2p} and B^{+}_{e2p}. Persian to English was best with CV-MODEL1, which outperformed 2\2T in improving the translation accuracy of high-ranked results. The main reason for CV-MODEL1 performing more accuracte for Persian to English is that in CV-MODEL1 the consonant-vowel sequences are segmented to CC patterns whereas in CV-

MODEL3 we cluster them in one segment. On the other hand, Persian words are mostly consonants. Hence, CV-MODEL1 helps to break the long sequences of consonants. Overall, We conclude that transliteration effectiveness can be improved using language features, even though they are shallow attributes of words such as consonants and vowels.

In this chapter, we also addressed the question of the effect of alignment on transliteration accuracy. We observed that when the alignment algorithm matched with the segmentation algorithm in our proposed methods, transliteration accuracy increased. By applying the same concept of consonant-vowel sequences on alignment, and with the aid of frequencies, our English-Persian transliteration effectiveness significantly improved.

The last contribution of this chapter was a novel back-transliteration method. We first examined current forward transliteration methods for the backward task to evaluate different transliteration methods examined for forward transliteration perform in the backward transliteration task. Unsatisfactory results gained by applying the previous methods on back-transliteration led us to approach a new algorithm for backward direction. We then modified these methods to create a back-transliteration system which generates English words from their transliterated Persian word by introducing reverse-segmentation. Reverse-segmentation is intended to recover the information that is lost in forward transliteration. The approach enhanced the overall accuracy obtained from CV-MODEL3, which incorporated both the new alignment algorithm and reverse-segmentation.

Chapter 5

Combining Automated Transliteration Systems

"Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away."

— Antoine de Saint-Exupery

Typically, most transliteration studies suggest specific source-target character alignment and word segmentation methods to form their transformation rules. However, the specific segmentation approach of a transliteration model can cause errors for some inputs, leading to incorrect transliteration. In this chapter, we investigate combining the outputs of multiple transliteration methods by majority voting and automatic classification schemes, keeping the internal function of each system intact. This approach takes advantage of the strengths of each model and reduces the impact of its weaknesses.

Aiming to improve transliteration effectiveness, we address the following two main problems to be investigated if the combination of transliteration systems should improve English-Persian and Persian-English transliteration efficacy:

1. which methods of system combination are useful for our task; and,
2. which transliteration systems should participate in the combination?

Glass-box combination has been investigated for transliteration in the literature (see

Section 2.3.3, page 46) but shows improvements only for specific tasks. Black-box combination has also been demonstrated to be effective for many applications, particularly machine translation and parsers (see Section 2.3.4, page 49). In this chapter we opt for black-box combination to preserve the nature of each transliteration model (as oppose to glass-box methods). We investigate several different *combination schemes* for integrating the individual models into a single system. Given the high number of possible *system combinations* of 15 different models we chose, we limit the combinations that we investigate by choosing the models that perform the best individually for combining into the final system. Our method is similar to the system proposed by Henderson and Brill [1999] for parsers, in which classifiers are applied to combine the output of multiple parsers.

We report our experiments on methods of combining the output of transliteration systems to create a single ranked list of target words with emphasis on improving the first suggested transliteration (TOP-1). Evaluating the proposed scheme for Persian-English and English-Persian transliteration, we observe either improvement or equal translation accuracy in comparison to using the output of individual systems.

5.1 Experimental Setup

Many different systems can be selected to participate in combination experiments. A number of them were explored in Chapter 3 and Chapter 4, including systems based on n-grams and consonant-vowel sequences. We chose a variety of n-gram based models that apply past or future context, and also three consonant-vowel based systems, fifteen models in total, for both English-Persian and Persian-English transliteration.

Two corpora are used in our experiments: the EDA_7 corpus, introduced in Chapter 2, for English to Persian transliteration, contains 1,500 word pairs transliterated by seven human transliterators. For Persian to English we use the B_{p2e} corpus of 2,010 Persian personal names. Each word in this corpus is accompanied by one to five transliteration variants.

We use uniform word accuracy in our evaluations which equally values all the transliteration variants of each source word. Our experimental results are reported for TOP-1, TOP-2 and TOP-5; these metrics are discussed in detail in Chapter 2.

5.2 System Combination

As explained in Section 5.1, many candidate transliteration models could be included in a combined scheme; for example all possible combinations of past and future context size could be used. Since we aim to maximise the final accuracy in a combined system, we are interested in selecting those systems that provide the most benefit. In this section, we compare the effectiveness of individual systems, and then analyse their outputs to determine which ones have the potential to most significantly boost the overall performance of a combined model.

5.2.1 Individual Systems

We used eleven n-gram based systems and three consonant-vowel based models on both English-Persian and Persian-English corpora. In addition, we used a system based on hand-crafted transliteration rules, we noted as MANUAL (described in Appendix D). The performances of these systems are summarised in Table 5.1.

The results for English-Persian transliteration suggest that CV-MODEL3 has the highest performance of all individual systems, with 74.3% TOP-1 word accuracy. From eleven reported n-gram based methods, 1\0 is the most effective system with 59.7% TOP-1 accuracy. Using a target language model always has a negative impact on the results, consistent with previous results reported in Chapter 3 for different corpora. Persian-English transliteration however is best achieved with CV-MODEL1, with 52.8% TOP-1 accuracy. In contrast to English-Persian transliteration, using a target language model does not always diminish the effectiveness; in some cases the effectiveness may even be increased. The collection of only personal names used for Persian-English might explain why less diversity in the string patterns of the target words is experienced in the target language model, and therefore for some systems better results are achieved. According to these experiments, Persian-English transliteration benefits more from both past and future context than just one past context in English-Persian. In general, there is no system that performs well on both English-Persian and Persian-English tasks.

	English-Persian			Persian-English		
SYSTEM	TOP-1	TOP-2	TOP-5	TOP-1	TOP-2	TOP-5
MANUAL	56.2	57.9	58.7	16.6	20.6	21.4
0\0	56.6	62.4	71.9	13.6	24.0	36.2
1\0	59.7	76.1	87.1	33.9	43.7	51.6
1\0T	35.0	41.3	48.5	21.9	27.5	41.0
2\0	37.0	57.7	70.4	12.4	22.5	32.5
2\0T	35.0	41.3	48.5	20.5	24.9	33.4
1\1	56.6	66.2	74.4	41.4	52.8	63.2
1\1T	31.3	34.0	49.3	38.8	51.6	64.4
2\2	57.4	66.4	73.9	42.2	52.6	61.2
2\2T	48.3	56.4	66.6	47.1	57.5	66.5
2\1	32.9	50.9	63.2	24.2	35.1	51.3
1\2	49.1	60.9	73.0	26.1	38.3	56.9
CV-MODEL1	59.9	73.9	84.4	52.8	63.4	72.7
CV-MODEL2	61.7	74.1	88.3	36.5	49.2	64.0
CV-MODEL3	74.3	85.7	93.8	39.0	52.6	68.0

Table 5.1: Mean uniform word accuracy of MANUAL*, n-gram based and consonant-vowel based transliteration systems. A* T *indicates that a target language model is used.*

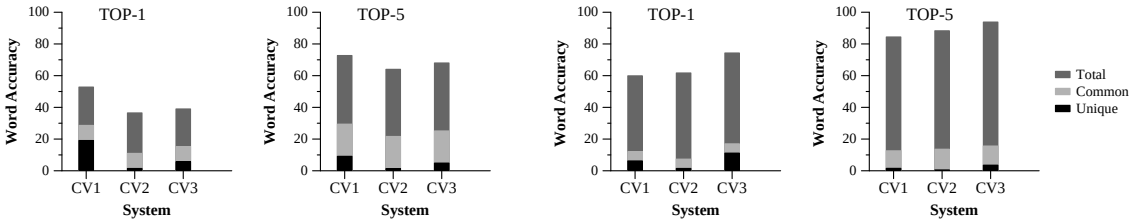
5.2.2 Analysis of Individual Outputs

Typically, the main difference between transliteration methods lies in their source word segmentation step, which leads to different transliterations or rankings. Although some systems show higher performance in terms of average accuracy, case by case analysis of their results reveals that they might fail in transliterating some words that are in fact successfully transliterated by generally weaker systems.

Let the accuracy of each system i be A_i. Then $A_i = A_{u_i} + A_c + A_i{}'$ where A_{u_i} is system i's contribution to accuracy by words that are correctly transliterated only by this system and no other ("Unique"); A_c is the contribution to accuracy by words that are correctly transliterated by all systems ("Common"); and $A_i{}'$ is the remainder (correctly transliterated by more than just system i, but less than all systems). The occurrence of common correct answers from more than one system inspires us to use a voting method to identify those systems which contribute to $A_c + A'$. For all names correctly transliterated by only one system, $A_u = \sum_i A_{u_i}$, we investigate the use of classifiers to attribute the "correct" system to a name.

Here, we analyse the percentage of unique correct transliterations of each system (A_{u_i}), the percentage of common correct transliterations (A_c), and the percentage of correct transliterations — the individual systems (A_i). In addition, the percentage of common incorrect transliterations — or common error (E_c) — is considered as an opposite factor to A_c. Since all possible combinations of the 15 systems listed in Table 5.1 are too numerous to report, we focus on consonant-vowel based systems (Figure 5.1) and a combination of all 15 systems (Figure 5.2). When the number of systems involved increases, the percentage of unique transliterations decreases. That is, if we have two schemes M_1 and M_2, where M_x is the set of systems involved, and $|M_2| > |M_1|$, then we expect that $A_{u_2} < A_{u_1}$ and $A'_2 > A'_1$. We are particularly interested in those transliterations that are common among all the systems. Our observations show that by increasing the number of systems, the number of unique transliterations drops, while there is an increase in common transliterations, both correct and incorrect. For example in Figure 5.1, where only three systems are involved, accumulated unique accuracy is $A_u = 26.2\%$ for Persian-English when only top-rank transliterations are considered. Increasing the number of systems to 15 reduces this value to 9.6%, as shown in Figure 5.2. This effect can also be seen in common incorrect transliterations, which drops

(a) Persian-English (b) English-Persian



$A_u = 26.2$ $A_c = 9.5$ $E_c = 12.7$

$A_u = 15.3$ $A_c = 20.3$ $E_c = 3.6$

$A_u = 3.4$ $A_c = 4.0$ $E_c = 7.9$

$A_u = 0.7$ $A_c = 4.3$ $E_c = 14.3$

Figure 5.1: Mean uniform word accuracy for the three consonant-vowel based systems individually (Total), the percentage of words uniquely transliterated correctly by each system (Unique), and the percentage of the words correctly transliterated by all the three systems (Common). Calculated A_u, A_c and E_c are given for each diagram.

from 12.7% to 0.6%, when using more systems. We should therefore use combinations of individual systems which minimise A_u and E_c and maximise A_c, thereby maximising the total accuracy.

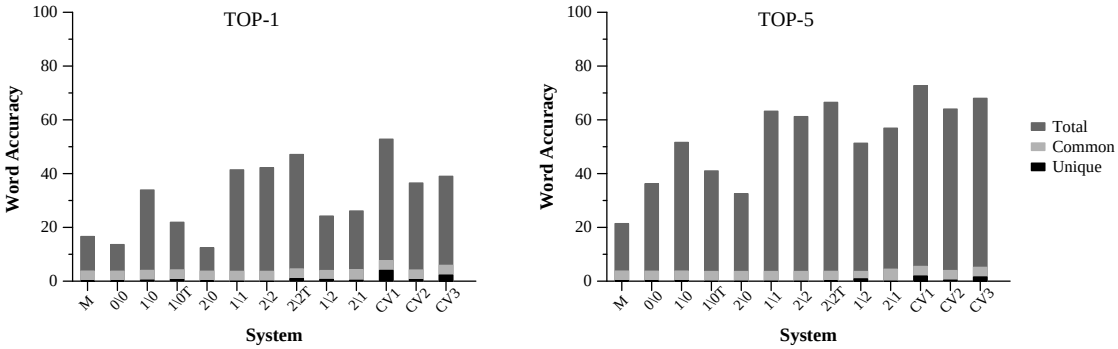
In general, as can be seen in Figures 5.1 and 5.2 and also the results listed in Table 5.1, when properly harnessed, useful features of multiple systems can be combined to achieve higher accuracy. Our main purpose here is to re-rank the output lists to move the correct transliteration up in the list, relying on the fact that most systems (except MANUAL) identify more correct answers in TOP-5 rather than TOP-1; for example, in English-Persian transliteration CV-MODEL3 correctly transliterates 93.8% of the test words in TOP-5 versus 74.3% in TOP-1; pushing the correct answer up to position one would give a 19.5% increase in TOP-1 accuracy.

We consider integrating the outputs of multiple transliteration systems in the next section.

5.3 Combination Schemes

With an available ranked list for each source word S from h different systems that are trained on identical data, we define two approaches of merging these lists: first, using the

(a) Persian-English

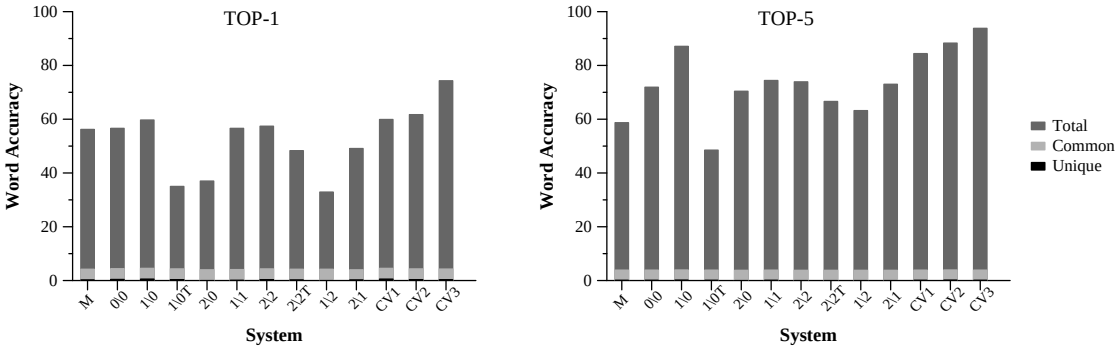


$A_u = 9.6$ $A_u = 5.1$

$A_c = 3.4$ $A_c = 3.9$

$E_c = 0.6$ $E_c = 2.6$

(b) English-Persian



$A_u = 3.4$ $A_u = 0.7$

$A_c = 3.9$ $A_c = 4.3$

$E_c = 0.1$ $E_c = 1.4$

Figure 5.2: Mean uniform word accuracy for 15 different systems individually (Total), the percentage of words uniquely transliterated correctly by each system (Unique), and the percentage of the words correctly transliterated by all the 15 systems (Common). Calculated A_u, A_c and E_c are given for each diagram.

frequency of suggested words (voting); and second, using the features of source and target words (classification). We also integrate these two approaches.

5.3.1 Majority Vote in Top-n (MVn)

Majority voting is defined as a voting scheme whereby the option with a majority of votes wins. Basically, it is a mapping that associates individual preferences with a resulting outcome using two variant selection outlines: *simple* and *absolute*. An *absolute majority voting* scheme guarantees that the vote in favour of the winner is significant; it implements a rule R^k which forces the number of votes in favour of winner be greater than the threshold k. On the other hand, *simple majority voting* selects any option that achieves more votes than others. Here, we adapt these concepts for our system output selection. Participating transliterators, a set of $M = \{M_1, M_2, \ldots, M_h\}$ systems, express their TOP-1 transliteration choice T_i as their vote to build a voting vector of $V = (V_1, \ldots, V_h)$. A voting rule maps the voting vector to any possible transliteration T of the source word. We define *absolute* and *simple* voting rules, R_a^k and R_s respectively, as follows:

$$R_a^k = \begin{cases} T_i & \text{if } f(T_i) > k \text{ where } k = h/2; \\ T_q & \text{otherwise.} \end{cases}$$

$$R_s = \begin{cases} T_i & \text{if } f(T_i) > f(T_j) \text{ and } i \neq j; \\ T_q & \text{otherwise.} \end{cases}$$

where the function $f(T_i)$ returns the frequency of T_i in V, and T_q is the default winner in case of majority failure. Pseudo-code for *simple majority voting* is presented in Algorithm 5.

As an extended approach, we let the systems vote n times. That is, each system provides TOP-N transliterations for a source word: $V = (V_{11}, \ldots, V_{1n}, \ldots, V_{h1}, \ldots, V_{hn})$. In this case, the same *absolute* voting rule ${R_a}^k$ applies, but the threshold is based on the number of votes available. When all systems actually have n suggestions to present, the threshold is $h \times n/2$. We refer to this voting system as MVn. In terms of merging the output lists, following each voting the output list L_i of the participant system i is updated to remove the winner transliteration T_i – anywhere in any list – effectively shifting the rest of the suggested transliterations up in the list.

Persian-English						
Combination Scheme	System Combination	TOP-1	TOP-2	TOP-5	diff	(p-value)
Baseline	CV-MODEL1	52.8	63.4	72.7	—	
Simple MV1	M_1	49.3	60.5	72.7	-3.5	(0.0211)
	M_2	55.8	67.5	76.2	+3.0	(0.0097)
	M_3	45.3	63.1	75.8	-7.5	(< 0.0001)
	M_4	52.5	62.4	72.2	-0.3	(0.7768)
	M_5	57.9	68.8	76.5	+5.1	(< 0.0001)
	M_6	52.0	65.6	76.9	-0.8	(0.4662)
Simple MV5	M_1	49.6	63.5	75.9	-3.2	(0.0821)
	M_2	55.9	69.6	79.8	+3.1	(0.0246)
	M_3	50.9	62.4	73.3	-1.9	(0.1026)
	M_4	56.6	65.8	74.0	+3.8	(0.0420)
	M_5	58.9	71.0	80.0	+5.7	(< 0.0001)
	M_6	54.2	68.8	78.6	+1.4	(0.2616)
Absolute MV1	M_1	53.2	64.8	74.1	+0.4	(0.2212)
	M_1	55.5	66.0	75.0	+2.7	(0.0005)
	M_2	55.8	67.5	76.0	+3.0	(0.0089)
	M_3	45.3	63.1	75.6	-7.5	(< 0.0001)
	M_4	53.6	64.4	74.3	+0.8	(0.4794)
	M_5	57.9	68.8	76.4	+5.1	(< 0.0001)
	M_6	54.1	66.2	75.2	+1.3	(0.1175)
Absolute MV5	M_1	52.3	67.2	76.8	-0.5	(0.6838)
	M_2	55.9	69.6	79.7	+3.1	(0.0246)
	M_3	50.9	62.4	73.3	-1.9	(0.1026)
	M_4	56.6	65.8	74.7	+3.8	(0.0984)
	M_5	58.9	71.0	80.0	+5.7	(< 0.0001)
	M_6	54.2	69.3	78.9	+1.4	(0.1956)

*Table 5.2: Mean uniform word accuracy (*UWA*) for the combined systems using majority voting for Persian-English. Where the* absolute *majority vote is applied and the threshold is not met, one vote from the best system (*CV-MODEL1*) is taken. The percentage of difference (diff) between baseline and the examined methods is given in the last column for* TOP-1 *results. System combinations are:* M_1=*{all 15 systems}*, M_2={CV-MODEL1, CV-MODEL3, 1\0, 2\2 }, M_3={CV-MODEL1, CV-MODEL2, CV-MODEL3 }, M_4={CV-MODEL1, CV-MODEL3, 1\1, 2\2, 2\2T}, M_5={CV-MODEL1, CV-MODEL3, 1\0, 2\2T} *and* M_6={CV-MODEL1, CV-MODEL2, CV-MODEL3, 1\0, 2\2 }.

English-Persian						
Combination Scheme	System Combination	TOP-1	TOP-2	TOP-5	diff	(p-value)
Baseline	CV-MODEL3	74.3	85.7	93.8	—	
Simple MV1	M_1	67.9	81.8	90.9	-6.4	(0.0001)
	M_2	71.9	84.4	93.0	-2.4	(0.0733)
	M_3	67.6	81.1	90.1	-6.7	(< 0.0001)
	M_4	71.5	82.7	90.1	-2.8	(0.0021)
	M_5	74.9	86.4	94.5	+0.6	(0.0246)
	M_6	73.3	83.5	91.4	-1.0	(0.0034)
Simple MV5	M_1	71.0	84.5	93.2	-3.3	(0.0213)
	M_2	75.7	87.4	94.5	+1.4	(0.2050)
	M_3	69.0	81.9	92.0	-5.3	(< 0.0001)
	M_4	72.9	84.9	93.5	-1.4	(0.2024)
	M_5	75.7	86.8	93.9	+1.4	(0.0990)
	M_6	73.2	85.6	94.5	-1.1	(0.0246)
Absolute MV1	M_1	63.5	76.5	88.7	-10.6	(< 0.0001)
	M_2	77.1	87.7	94.7	+2.8	(0.0034)
	M_3	71.7	86.1	94.4	-2.6	(0.0038)
	M_4	73.7	86.0	94.5	-0.6	(0.1945)
	M_5	75.2	86.8	94.2	+1.4	(0.1808)
	M_6	74.9	86.4	94.5	+0.6	(0.0246)
Absolute MV5	M_1	69.5	83.7	92.5	-4.8	(0.0019)
	M_2	76.1	88.3	95.2	+1.8	(0.0911)
	M_3	69.3	82.1	92.8	-5.0	(< 0.0001)
	M_4	73.9	86.5	94.5	-0.4	(0.6586)
	M_5	76.0	87.3	94.5	+1.7	(0.0990)
	M_6	73.2	85.6	94.5	-1.1	(0.0246)

Table 5.3: Mean uniform word accuracy for the combined systems using majority voting for English-Persian. Where the absolute *majority vote is applied and the threshold is not met, one vote from the best system (*CV-MODEL3*) is taken. The percentage difference (diff) between the baseline and the examined methods is given in the last column for* TOP-1 *results. System combinations are:* M_1=*{all 15 systems},* M_2={CV-MODEL1, CV-MODEL3, 1\0, 2\2 }, M_3={CV-MODEL1, CV-MODEL2, CV-MODEL3 }, M_4={CV-MODEL1, CV-MODEL3, 1\1, 2\2, 2\2T}, M_5={CV-MODEL1, CV-MODEL3, 1\0, 2\2T} *and* M_6={CV-MODEL1, CV-MODEL2, CV-MODEL3, 1\0, 2\2 }.

Algorithm 5 Majority vote system selection.

1: Let h be the number of systems involved.
2: Let the ranked list L_i be the output of system i.
3: Let *head*(L_i) return the first item in L_i.
4: Let q be the most effective individual system.
5: **while** there is an unprocessed target word in any L_i list **do**
6: $V = (head(L_1), \ldots, head(L_h))$.
7: Add T, the most frequent element of vector V, to L. If all elements in V have the same frequency, choose element q.
8: Remove T from all the output lists L_i.
9: **end while**
10: Output L.

5.3.2 Classification-based System Selection

Classification and clustering have been previously applied for transliteration to group input terms based on their type, for example into classes of personal names and organisation names [Chen and Lee, 1996], or language of origin of the source words [Huang et al., 2005; Huang, 2005]. In such studies a transliteration model is formed based on the classified input. By contrast, we consider classification as a method of selecting the output of transliteration systems. That is, our approach isolates the training stage of the transliteration models from the classifier.

In our experiments we applied a *Naïve-Bayes classifier* [Duda and Hart, 1973; Mitchell, 1997]. Naïve-Bayes classification works based on the assumption of independence of attributes we consider in classification task, and is generally considered to be a competitive method of text classification. This classifier has also been investigated for other combination studies, such as parsers [Henderson and Brill, 1999] where it has been reported a successful method. This classifier follows two steps: training and classification. Details on how it is employed in our transliteration task are explained below.

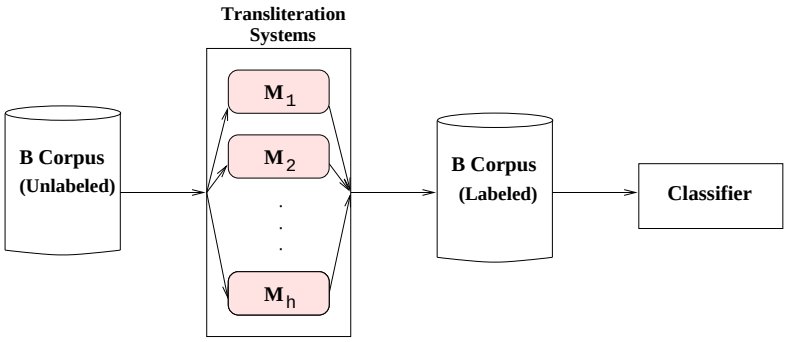


Figure 5.3: Process of generating a labelled corpus for training the classifier using the output of all the transliteration systems.

Training Stage

Each source word S in the training data set might be correctly transliterated by one or more than one system (if none, it is not counted in the training step). For every training instance $< (S,T), M >$, where M is the set of transliteration methods which transliterate the source word S correctly, we specify the corresponding features (for example n-grams) of the pair (S,T). Figure 5.3 shows the flow of training for such a classifier in our proposed system. When training and testing of the entire corpus for *transliteration* is accomplished, the data can be labelled based on those systems which were able to correctly transliterate the source words. Such data can then be utilised for the training stage of the *classifier*. In our system, we use three sets of features: first, n-grams of the actual words (source and target); second, n-grams of consonant-vowel sequences of the source word; and third, word length. We present experiments with different combinations of features below.

Classification Stage

Given a test word pair $w = (S,T)$ with the feature set $\{f_1, f_2, \ldots, f_n\}$, the classification probability is computed as follows:

$$\begin{aligned} P(M|w) &= P(w|M).P(M)/P(w) \\ &\propto P(f_1, f_2, \ldots, f_n|M).P(M) \\ &= P(M).\textstyle\prod_{j=1}^{n} P(f_j|M) \\ M_w &= \text{argmax}_{M_i \in M} P(M_i|w) \end{aligned}$$

where M represents a class (system) and $P(M|w)$ is the posterior probability of allocating the seen data w to the class M. We assume prior probability of $P(M)$ is uniform.

Estimation of sparse multinomial distributions is an important component of many statistical learning tasks, such as text classification. The most straightforward method of calculating the posterior probabilities uses the frequency of allocating an attribute to a class as n_f/n_M, where n_f is the frequency of allocating feature f to class M, and n_M is the total number of times any feature is assigned to the class M. Such a probability calculation can easily lead to zero values when $n_f = 0$, that is, when the attribute f is not seen in the training data. Many smoothing methods have been proposed to solve this zero frequency problem [Jurafsky and Martin, 2008]. A simple, yet useful method of smoothing for *Naïve-Bayes classifier* is m-estimation in which probabilities are calculated as:

$$P(f|M) = \frac{n_f + mp}{n_M + m}.$$

The parameter m is called the *equivalent sample size*, which augments the number of actual observations with m additional virtual samples, and p is prior estimate of the probability $P(f|M)$.

In document classification the Laplace approach is also widely used [Mitchell, 1997], this applies the same formula as m-estimation with $mp = 1$ and $m = |\text{vocabulary}|$:

$$P(f|M) = \frac{n_f + 1}{n_M + |\text{vocabulary}|}.$$

We substitute $|\text{vocabulary}|$ with $|F|$, where F is the set of all the features seen in the training data.

Pseudo-code for system selection based on classification is shown in Algorithms 6 and 7 demonstrating the training and classification phases, respectively. In Algorithm 6 probability estimation is presented using m-estimation, with uniform priors that set $p = 1/h$ with h equal to the number of systems (classes) [Mitchell, 1997]. Algorithm 7 begins with posterior probabilities from the training stage calculated using Algorithm 6 , and generates an output list from ranked lists provided by participating transliteration systems.

Algorithm 6 Classification based system selection: training stage.

Let $M = \{M_1, M_2, \ldots, M_h\}$ represent all the possible classes, with each class representing a transliteration system with output list L_i.
Let n_f represent the frequency of the feature f.
Let m be the equivalent sample size, and $p = 1/|M|$.
for each $M_i \in M$ **do**
  for each training word-pair (S, T) **do**
    Extract all the desired features of (S, T) as a set F_i.
  end for
  for each $f \in F_i$ **do**
    Calculate $P(f|M_i) \leftarrow \frac{n_f + mp}{|F_i| + m}$.
  end for
end for
Output all the $P(f|M_i)$ probabilities.

5.4 Experiments and Discussion

To examine the effect of combining transliteration systems, we ran three sets of experiments. The first set evaluates the effect of majority voting using both *simple* and *absolute* approaches; the second set explores the impact of applying different features on a classifier and its impact on a system that selects the output of individual transliteration systems; and the third set of experiments evaluates combination schemes that integrate majority voting and classification approaches.

5.4.1 Effect of Majority Voting

We examined the efficacy of *simple* and *absolute* voting on the selection of systems; results are reported in Table 5.2 and Table 5.3 for Persian-English and English-Persian, respectively. The best system from Table 5.1 was considered as a *default* winner when there is no majority. The best choice for Persian-English is CV-MODEL1 and for English-Persian it is CV-MODEL3, therefore they are selected as the baseline for our comparisons, and also as default systems. We report a selection of different system combinations in Tables 5.2 and 5.3, denoted as M_1

Algorithm 7 Classification based system selection: classification stage.

Require: Posterior probabilities ($P(f_j|M_i)$).

Let $M = \{M_1, M_2, \ldots, M_h\}$ represent all the possible classes, with each class representing a transliteration system with output list L_i.
Initialise $L \leftarrow \epsilon$ as an empty final output list.
Let *head(L_i)* return the first item in L_i.
Let q be the most effective individual system.
Initialise $P_{\max} \leftarrow 0$.
for each testing word-pair (S, T) **do**
  while there is an unprocessed target word in any L_i list **do**
    $d \leftarrow q$.
    for each $M_j \in M$ **do**
      Extract all the desired features of (S, T) as a set of $F_i = \{f_j\}$.
      $P((S,T)|M_i) \leftarrow \prod_j P(f_j|M_i)$.
      if $P((S,T)|M_i) > P_{\max}$ **then**
        $P_{\max} \leftarrow P((S,T)|M_i)$.
        $d \leftarrow i$.
      end if
    end for
    Add *head(L_d)* to the end of L.
    Remove T from all output lists L_i.
  end while
end for
Output L.

to M_6, as a representative subset of many different combinations that could be selected.

The best set for Persian-English was M_5 ={CV-MODEL1, CV-MODEL3, 1\0, 2\2T} with 58.9% TOP-1 accuracy and standard deviation of 2.3, performing 5.7% (absolute difference) better than the baseline system using *simple* or *absolute* MV5. The improvement is statistically significant (paired *t-test*, $p < 0.0001$).

For English-Persian the best combination is the set M_2 ={CV-MODEL1, CV-MODEL3, 1\0, 2\2 }, giving uniform word accuracy of 77.1% TOP-1 with standard deviation of 4.2, a 2.8% absolute improvement over the baseline using *absolute* MV1. The best sets of systems we examined are M_2 and M_5 in which *simple* and *absolute* voting methods perform identically for Persian-English. For English-Persian on the other hand, *absolute* voting is generally more effective; however, for M_2 and M_5 no significant difference is observed between *simple* and *absolute* MV5. More precisely, using such combinations, all the integration methods perform equally when transliterating from Persian to English, and if English to Persian transliteration is required these combinations give superior performance to other sets especially when *absolute* MV5 is used. One reason for the success of such system combinations is that they are selected from the three distinct categories of systems we had (listed in Table 5.1): using past context only, using both past and future, and consonant-vowels based models. We use M_2 in our next set of experiments as our selected system combination.

5.4.2 Effect of Classifier

The results of using our classification approach, as explained in Section 5.3.2, are reported in Table 5.4. The integrated system selects outputs based on the classification decision. If classification was unsuccessful (probabilities of selecting all systems are equal or less than a defined threshold determined emprically as explained later in Figure 5.5), it uses the output of the single system that gave the best performance during training for each language pair (the same default systems of Experiment Set 1). The features used are source word n-grams (SNG), source word consonant-vowel sequences (SCV), source word length (SWL), and target language n-grams (TNG). We also investigated the effect of two smoothing options, m-estimation and Laplace, for all the examined features.

Our experiments suggest that just using source word n-grams is as effective as other feature choices. The two smoothing paradigms did not show much difference, with Laplace

smoothing giving slightly better performance. The best results for Persian-English was 66.4% TOP-1 accuracy with standard deviation of 6.1, significantly improving the baseline by 13.6% in absolute terms ($p < 0.0001$). For English-Persian, the best result was 88.1% TOP-1 with standard deviation of 6.0, improving the baseline by absolute 13.8% ($p < 0.0001$).

We also examined the effect of length of source n-grams separately, as shown in Figure 5.4. For both Persian-English and English-Persian increasing the length, from unigrams to 5-grams, had a positive effect. The benefit remained stable when adding more context up to 8-grams. We therefore report all the experiments using 5-grams, the best trade-off between performance and gram length.

5.4.3 Effect of the Combined Voting and Classification

We conducted experiments with combined systems that incorporate both majority voting and classification. A Combined system applies one approach (either majority voting or classification) first. If the first approach does not meet a required threshold, the other approach is used as a backoff strategy. We denote the combination schemes as NB+MV or MV+NB in Table 5.5 for Persian-English and Table 5.6 for English-Persian with the first named system being the first applied. For *absolute majority voting* the voting rule is ${R_a}^{h/2}$ with h representing the number of systems involved. For classification, threshold P_{max} in Algorithm 7 is set to 10^{-20}. The threshold is determined empirically as shown in Figure 5.5. Transliterating from English to Persian using two combination schemes, classification-only and classification plus majority voting, we varied the threshold on probability of choosing a system using a classifier or another method (either default system or voting) from 10^{-3} to 10^{-35}. In the range of thresholds greater than 10^{-15} and less than 10^{-25} the best effectiveness using both NB only and NB+MV schemes was achieved.

Systems with NB as the first option outperformed the others with majority voting as the first selection method; the combination results in a further improvement to each selection scheme. The most effective scheme for both Persian-English and English-Persian were NB+MV1 and NB+MV5 with 16.7% (< 0.0001) and 14.2% (< 0.0001) absolute improvement, respectively, when the selected system combination M_2 was applied. Given that there is no difference in using any of the examined voting methods for our best combination sets, similar results are achieved for all the NB+MV integration methods reported in Tables 5.5

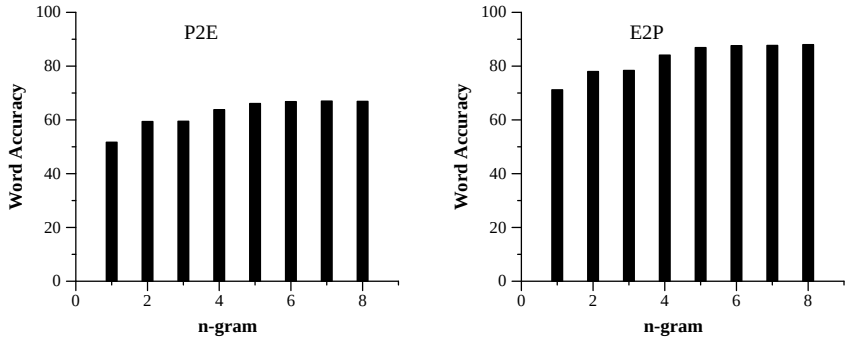


*Figure 5.4: Effect of length of features (n-gram) on the accuracy (*TOP-1*) of classification-based combined systems.*

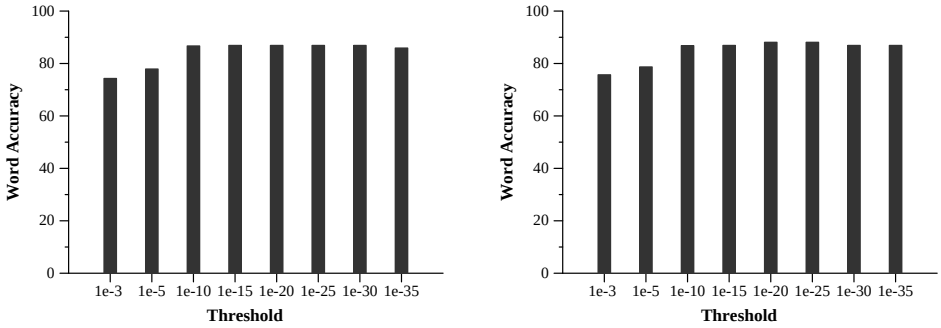


Figure 5.5: Effect of threshold for classification-based systems. The left diagram shows system performance in terms of uniform word accuracy using NB classifier with best single-system as default. The right diagram uses NB+MV1 (absolute). Both diagrams are reported for English to Persian transliteration.

and 5.6. Another reason for very similar results reported in these tables is that when the classifier is applied prior to a voting scheme, NB dominates the selection process, leaving fewer inputs undetermined (less than 4.0% in our experiments). With a system combination that applies all the systems listed in Table 5.1, NB+MV still outperforms MV+NB methods. In such a combination, improvements are obtained in MV+NB approaches only if *absolute* voting is used.

5.5 Conclusions

Many different transliteration approaches can be used to transliterate words from one language to another, each of them having different strengths and weaknesses. Our analysis of which systems generate correct output showed that even those systems that perform poorly on average can sometimes find unique correct answers for source terms on which other systems fail. Motivated by this observation, we investigated and experimentally evaluated a number of approaches for combining the output of different transliteration systems, aiming to draw on the strengths of individual systems to boost overall performance. Majority voting, where results are re-ordered based on their frequency of occurrence in the output of individual systems, leads to a 5.7% absolute improvement over the best single system baseline for Persian-English transliteration, and a 2.8% improvement for English-Persian transliteration. Classification approaches using a Naïve-Bayes classifier equipped with a Laplace smoothing approach, where systems are chosen based on n-grams features, gives an absolute improvement of 13.6% and 13.8% for Persian-English and English-Persian, respectively. Combining the majority voting and classification approaches shows even greater benefits; the most effective combinations first apply classification, and use majority voting as a back-off scheme, leading to absolute improvements of 16.7% for Persian-English and 14.2% for English-Persian. The absolute improvements translate to 31.6% and 19.1% relative improvements in performance respectively. Our novel techniques lead to significant improvements over highly effective single-system transliteration baselines.

Based on these findings, we can now answer the research problems mentioned earlier in this chapter. The improvements in transliteration accuracies observed in the experiments positively answer the question on whether black-box combination helps transliteration. We also addressed the problem of choosing useful methods of system combination for our task. We investigated classification and majority voting and both showed improvements under special system combinations. A combination of these two approaches was also examined and the best results were reported with a scheme that applied a classifier prior to majority voting. The last problem was: which transliteration systems should participate in the combination? This question is closely related to the previous problem of combination method selection and was examined through the same sets of experiments by differing the sets of systems participating in the combinations. We found the best set of participants that maximised the

effectiveness of the final system, which was composed of two systems from each group of the methods based on n-grams and consonant-vowel sequences.

Persian-English

Combination Scheme	Features	TOP-1	TOP-2	TOP-5	diff (p-value)
Baseline	—	52.8	63.4	72.7	—
	SNG	65.9	70.4	75.7	+13.1 (< 0.0001)
NB	SNG+SWL	65.9	71.4	77.8	+13.1 (< 0.0001)
(m-estimation)	SNG+SWL+SCV	66.0	71.5	78.9	+13.2 (< 0.0001)
	TNG	60.4	66.6	73.4	+7.6 (< 0.0001)
	all above	65.1	71.6	76.8	+12.3 (0.0008)
	SNG	66.4	72.1	78.0	+13.6 (< 0.0001)
NB	SNG+SWL	66.3	71.6	77.5	+13.5 (< 0.0001)
(Laplace)	SNG+SWL+SCV	66.3	71.7	77.6	+13.5 (< 0.0001)
	TNG	56.7	62.2	68.4	+3.9 (0.0002)
	all above	59.4	63.1	68.3	+6.6 (0.0001)

English-Persian

Combination Scheme	Features	TOP-1	TOP-2	TOP-5	diff (p-value)
Baseline	—	74.3	85.7	93.8	—
	SNG	86.9	88.9	92.6	+12.6 (< 0.0001)
NB	SNG+SWL	87.0	91.0	94.6	+12.7 (< 0.0001)
(m-estimation)	SNG+SWL+SCV	87.0	90.9	94.6	+12.7 (< 0.0001)
	TNG	78.8	86.1	92.0	+4.5 (0.0114)
	all above	86.3	90.6	95.5	+12.0 (< 0.0001)
	SNG	88.1	91.7	94.9	+13.8 (< 0.0001)
NB	SNG+SWL	86.1	89.9	93.9	+11.8 (< 0.0001)
(Laplace)	SNG+SWL+SCV	85.7	89.4	93.7	+11.4 (< 0.0001)
	TNG	76.6	84.5	90.9	+2.3 (0.1377)
	all above	85.0	89.5	93.9	+10.7 (0.0001)

*Table 5.4: Mean uniform word accuracy (*UWA*) for a combination scheme using Naïve-Bayes (NB) classifier. The classifier is trained on the following features: unigram to 5-grams for both source (SNG) and target words (TNG), consonant-vowel sequence of source word (SCV) and its length (SWL). Smoothing parameters for m-estimation method (Algorithm 6) are set to $m = 20$ and $p = 1/|M|$. The system combination is $M = \{$*CV-MODEL*1,CV-MODEL*3,1\0,2\2$\}$ for both Persian-English and English-Persian.*

Persian-English						
Combination Scheme	System Combination	TOP-1	TOP-2	TOP-5	diff	(p-value)
Baseline	CV-MODEL1	52.8	63.4	72.7	—	
Simple MV1+NB	M_1	50.9	62.1	73.2	-1.9	(0.2196)
Simple MV5+NB	M_1	53.3	64.8	75.5	+0.5	(0.7820)
Absolute MV1+NB	M_1	54.9	67.1	77.2	+2.1	(0.0291)
Absolute MV5+NB	M_1	55.6	69.4	81.1	+2.8	(0.0213)
NB+Simple MV1	M_1	66.9	71.3	77.9	+14.1	(< 0.0001)
NB+Simple MV5	M_1	67.0	76.2	78.9	+14.2	(< 0.0001)
NB+Absolute MV1	M_1	66.1	71.3	77.9	+13.3	(< 0.0001)
NB+Absolute MV5	M_1	66.9	76.2	78.2	+14.1	(< 0.0001)
Simple MV1+NB	M_2	61.7	72.6	80.3	+8.9	(< 0.0001)
Simple MV5+NB	M_2	57.7	71.7	81.3	+4.9	(0.0094)
Absolute MV1+NB	M_2	61.7	72.6	80.3	+8.9	(< 0.0001)
Absolute MV5+NB	M_2	57.7	71.7	81.3	+4.9	(0.0094)
NB+Simple MV1	M_2	68.3	73.4	79.7	+15.5	(< 0.0001)
NB+Simple MV5	M_2	68.5	73.5	79.7	+15.7	(< 0.0001)
NB+Absolute MV1	M_2	69.5	72.9	78.5	+16.7	(< 0.0001)
NB+Absolute MV5	M_2	68.5	73.6	80.5	+15.7	(< 0.0001)

*Table 5.5: Mean uniform word accuracy (*UWA*) for the combined systems (majority voting and classifier). System combinations are* $M_1 = \{$*all 15 systems*$\}$ *and* $M_2 = \{$CV-MODEL1,CV-MODEL3,1\0,2\2$\}$.

English-Persian						
Combination Scheme	System Combination	TOP-1	TOP-2	TOP-5	diff	(p-value)
Baseline	CV-MODEL3	74.3	85.7	93.8	—	
Simple MV1+NB	M_1	67.9	81.3	90.8	-6.4	(< 0.0001)
Simple MV5+NB	M_1	70.7	84.5	93.3	-3.6	(0.0069)
Absolute MV1+NB	M_1	75.7	85.9	94.1	+1.4	(0.1728)
Absolute MV5+NB	M_1	77.3	86.7	95.0	+3.0	(0.0447)
NB+Simple MV1	M_1	84.9	88.3	91.6	+10.6	(0.0051)
NB+Simple MV5	M_1	85.1	91.7	92.2	+11.3	(0.0006)
NB+Absolute MV1	M_1	84.5	88.1	91.5	+10.2	(0.0101)
NB+Absolute MV5	M_1	85.6	91.5	92.1	+11.3	(0.0115)
Simple MV1+NB	M_2	80.4	89.9	95.3	+6.1	(0.0003)
Simple MV5+NB	M_2	76.1	87.4	94.8	+1.8	(0.1073)
Absolute MV1+NB	M_2	80.4	89.9	95.3	+6.1	(0.0003)
Absolute MV5+NB	M_2	76.1	87.4	94.8	+1.8	(0.1073)
NB+Simple MV1	M_2	86.3	90.6	94.5	+12.0	(< 0.0001)
NB+Simple MV5	M_2	88.1	91.6	95.9	+13.8	(< 0.0001)
NB+Absolute MV1	M_2	86.3	90.6	94.6	+12.0	(< 0.0001)
NB+Absolute MV5	M_2	88.5	91.7	96.2	+14.2	(< 0.0001)

*Table 5.6: Mean uniform word accuracy (*UWA*) for the combined systems (majority voting and classifier). System combinations are* $M_1 = \{$*all 15 systems*$\}$ *and* $M_2 = \{$CV-MODEL1,CV-MODEL3,1\0,2\2$\}$.

Chapter 6

Corpus Effects on Transliteration Evaluation

"Part of the inhumanity of the computer is that, once it is competently programmed and working smoothly, it is completely honest."

— Isaac Asimov

Just as in the previous chapters, most automatic transliteration systems employ a corpus of known source-target word pairs to train their systems, and typically evaluate their systems on a similar corpus. In this chapter the performance of transliteration systems on a controlled corpus is explored. In particular, the number, and prior language knowledge of human transliterators used to construct the corpus, and the origin of the source words that make up the corpus is controlled. Experiments show that the word accuracy of machine transliteration systems can vary by up to 30% depending on the corpus on which they are run. To prevent incorrect judgments over different systems, we recommend five human transliterators be involved in corpus construction to keep the ranking and perceived accuracy of the systems stable over different corpora.

6.1 Introduction

The performance of transliteration approaches are evaluated using a bilingual transliteration corpus (explained in Definition 5 on page 33). Traditionally, these pairs are either extracted from bilingual documents or dictionaries [AbdulJaleel and Larkey, 2003; Bilac and Tanaka, 2005; Knight and Graehl, 1998; Oh and Choi, 2006; Zelenko and Aone, 2006], or gathered explicitly from human transliterators [Al-Onaizan and Knight, 2002a; Zelenko and Aone, 2006]. Some evaluations of transliteration methods depend on a single unique target word for each source word, while others take multiple transliterations for a single source word into account.

The effects of corpus composition on the evaluation of transliteration systems has not been specifically studied, with only implicit experiments or claims made in the literature such as introducing the effects of different transliteration models [AbdulJaleel and Larkey, 2003], language families [Lindén, 2005] or application based (CLIR) for evaluation [Pirkola et al., 2006]. In this chapter, we first analyse the results of the experiments reported in Chapter 3 to investigate the reasons for experiencing a large gap between the performance achieved using two different English-Persian corpora. Based on these initial experiments on the testing and training corpora, we conclude that the language of origin of the source words heavily influences the experienced efficacy of the systems. We then expand those initial investigations by reporting our experiments designed to explicitly examine the effect of varying the underlying corpus used in both training and testing systems on transliteration accuracy. Specifically, we vary the number of human transliterators that are used to construct the corpus; and the origin of the English words used in the corpus. The aim of these experiments is to examine what conditions are fair for judging different transliteration systems against each other, especially when same evaluation corpus is not available.

In particular, we address the research questions on transliteration evaluation:

1. how many source words should a transliteration corpus consist of for a robust performance;
2. does the number of valid transliteration variants included in a corpus affect the comparison of performance of multiple systems; and,
3. does the origin of the source words provided to transliterators affect the perceived

accuracy of the transliteration systems?

The experiments, designed to answer the questions above, also revealed that the word accuracy of automated transliteration systems can vary by up to 30%, depending on the corpus employed. Despite the wide range of absolute values in performance, the ranking of the transliteration systems (best to worst) was preserved on all corpora. We also find that a transliterator's confidence in the language from which they are transliterating can affect the corpus in such a way that word accuracy rates are altered.

6.2 Analysis of the Difference of the Single-Transliterator Corpora

In Chapter 3, we compared the accuracy of systems based on various n-gram models. These experiments demonstrate which system more accurately transliterates out-of-dictionary words that appeared in our bilingual corpora. The difference of accuracies perceived for the same language pair of English-Persian with two corpora (B_{e2p} and B^{+}_{e2p}), however, is not explained. For example, despite the fact that B_{e2p} is a subset of B^{+}_{e2p}, 0\1 on B_{e2p} corpus yields 53.1% TOP-1 accuracy, but only 9.8% on B^{+}_{e2p}. In this section we report our experiments that explain the parameters causing differences on system accuracies for the B_{e2p} and B^{+}_{e2p} corpora. Specifically, the effect that language of origin of source words and the size of a corpus are examined.

6.2.1 Language-separated Corpora

As previously explained in Section 2.5 on page 58, the corpus construction process undertaken for B^{+}_{e2p} did not guarantee the origin of source words to be English. Many of them, although taken from English Web pages, have different language origins and were already transliterated once to English. Hence, B^{+}_{e2p} contains sub-collections of words of different languages. To examine their effect on the observed performance of individual transliteration systems, some of them are extracted from B^{+}_{e2p} forming the following four sub-collections: Dutch, Arabic, Indian and Chinese; containing 615, 254, 197 and 269 word pairs respectively. A source word was assigned to a country according to its appearance on Web pages of that country. For example, "Groenendijk" appears in many pages of the domain `.nl`, and so was assigned to the Dutch category.

All the country-based sub-collections were treated as separate corpora, on which 1\0 was run. The results, using mean word accuracy over 10-fold cross-validated collections, are shown in Figure 6.1. In this figure the mean word accuracy of B_{e2p} and B^{+}_{e2p} corpora are shown as vertical lines on the accuracy axis. Another collection was also made, concatenating all these country-based collections, called *Mixed*. The accuracy achieved for this mixed collection and the average accuracy of all separate ones (Mean) is also indicated as vertical lines. It can be seen that the accuracy is substantially reduced for all languages individually and when mixed. The accuracy of the *Mixed* collection is even less than the average (*Mean*) accuracy value of these corpora. Therefore, terms from languages other than English which are already transliterated — at least once — (or over-transliterated terms) have a negative effect on efficacy achieved from B^{+}_{e2p}. However, the sizes of these collections are small and this can be the reason for poor results; it could be raised that they could be transliterated more precisely if they were trained well. B^{+}_{e2p}, however, contains only small proportions of these sub-collections which makes the experiment valid for justifying the difference of B_{e2p} and B^{+}_{e2p}.

A closer investigation of what happened to over-transliterated words of B^{+}_{e2p} during transliteration makes this phenomenon more clear. We extracted the failure statistics of the words belonging to the country-based collections and their contribution to total failure of transliteration (Table 6.1). Arabic and Dutch words are rarely transliterated correctly in TOP-1 results, showing high failure rate for all the names involved in each test dataset.

Another view from the country-based collections is given in Figure 6.2, where the frequency distribution of target language alphabet, as the characters appear in rules in the trained model, is shown. Languages which are less accurately transliterated have a wider distribution over the target language alphabet. In Arabic, for instance, diversity is much greater than for Indian. Thus, the more variety that exists in picking the characters in transliteration of words, the more training data might be required to model that language accurately.

6.2.2 Corpus Scale

A general belief in machine learning tasks suggests that the larger the training samples, the better the resulting model for handling unseen samples. However, the experiments on

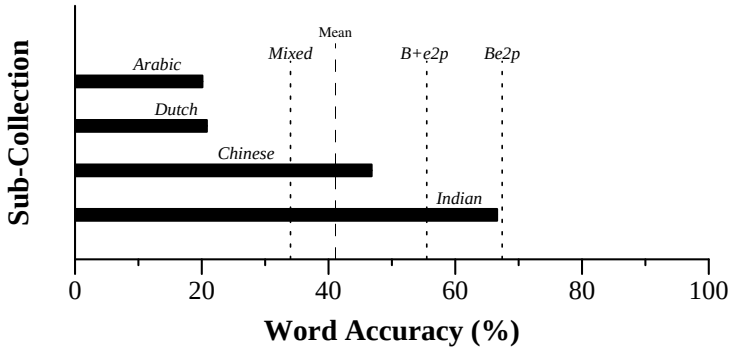


Figure 6.1: Mean word accuracy (TOP-1 USING SYSTEM $1\backslash 0$) *of country-based corpora extracted from* B^{+}_{e2p}.

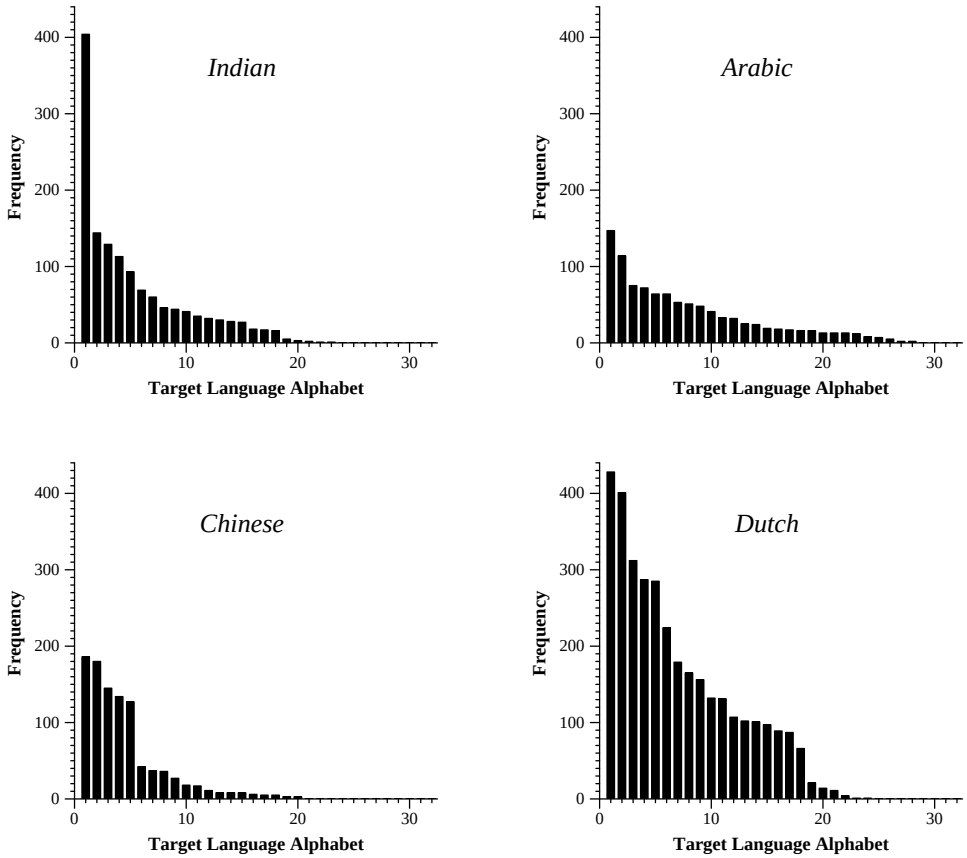


Figure 6.2: Distribution of usage of target language alphabet (Persian with alphabet size of 32) in transliteration of country-based corpora.

	Number of words	Incorrect Transliteration	Failure (%)
Arabic	5.6	4.2	77.7
Dutch	19.8	12.2	61.9
Chinese	10.0	3.3	32.4
Indian	6.3	1.9	31.0

Table 6.1: Statistics of transliteration failure for words with different language origins from B^{+}_{e2p}*. English-Persian transliteration is performed using* $1\backslash 0$ *and the failure results are averaged over* 10 *runs where each test corpus contains* $1,670$ *words of* TOP-1 *results.*

B_{e2p} and B^{+}_{e2p} do not confirm such an idea. B_{e2p} contains only 1,857 word-pairs whereas B^{+}_{e2p} consists 16,760 word-pairs. On the other hand the transliteration accuracy obtained on B_{e2p}, using any system, is remarkably higher than for B^{+}_{e2p}. We therefore investigated the effect that the size of a corpus may have on transliteration, and examined the minimum size required for such a corpus. We randomly partitioned B_{e2p} and B^{+}_{e2p} into sets of increasing size and computed the accuracy of $1\backslash 0$ on those partitions. The resulting diagrams are shown in Figure 6.3. For B_{e2p}, there are fluctuations in accuracy of corpora smaller than 500 word-pairs, while after this threshold, changes in performance slow down dramatically. These results suggest that any experiment performing on corpora of single language origin of size 500 word pairs or more is valid in their context. To further investigate this, the experiment was repeated for B^{+}_{e2p} to check our claim with a noisy, hard to model corpus. As predicted, the stable state of transliteration appears with the corpus of size 2000 word pairs, four times bigger than what was needed for B_{e2p} corpus. Increasing the corpus size, $1\backslash 0$ shows a slightly downwards trend especially for TOP-10 after a peak of corpus size in the range of 2000-5500. We thereby addressed the research question: how many source words should a transliteration corpus consist of for a robust performance?

The experiments suggest that one possible reason for poor performance of n-gram based systems on the B^{+}_{e2p} corpus is that it contains small sub-collections of non-English origin words. On the other hand, our experiment on B_{e2p} which contains words only of English origin shows that any data set above 500 words performs well. Hence, the small size of the individual country datasets is not to blame if their training requirement is similar to English

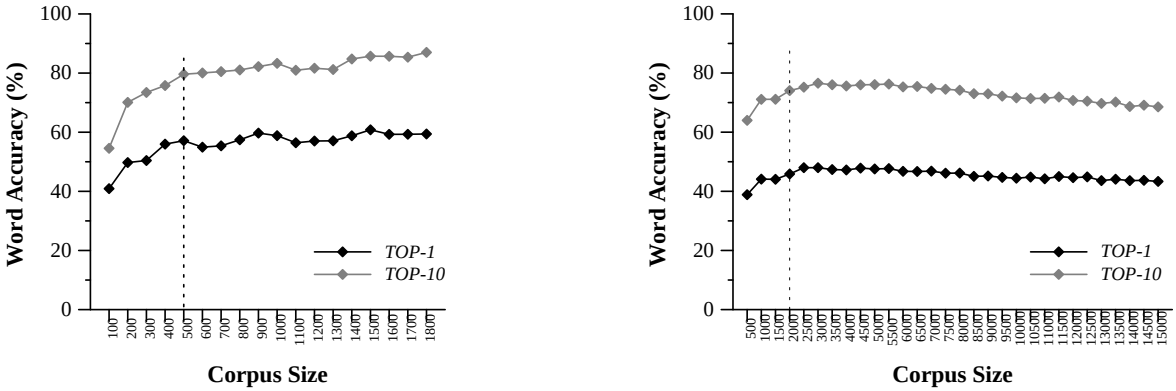


Figure 6.3: Mean word accuracy of corpora with different sizes for B_{e2p} (left) and B^{+}_{e2p} (right) using 1\0*. Lines are added to better show the trend and do not represent data points.*

words. Motivated by these initial results, we investigate this dilemma thoroughly with more carefully designed experiments, and a carefully constructed corpus that allows us to fully examine such effects.

6.3 Effect of Evaluation Scheme

In this section we compare the effect of corpus and evaluation metric on the evaluation results and system comparison. Two systems were compared in all the experiments: 1\0 and CV-MODEL3. The first system is representative for the best n-gram based E2P transliteration system according to the experiments in Chapter 3, and the second is the best consonant-vowel based E2P transliteration system as shown in Chapter 4.

To investigate the effect of choosing corpora with different origin on values we get from different metrics (UWA, MWA and WWA), the two selected systems are run over E_7, A_7, D_7 and EDA_7. The results are shown in Figure 6.4. Immediately obvious is that varying the corpora (x-axis) results in different values for word accuracy, whether by the UWA, MWA or WWA metric.

For example, if CV-MODEL3 was evaluated with the UWA metric on the D_7 corpus, a result of 82% would be obtained, but if you chose to evaluate it with the A_7 corpus you would receive a result of only 73%. This makes comparing systems that report results obtained on different corpora very difficult. Encouragingly, however, CV-MODEL3 consistently outperforms 1\0 on

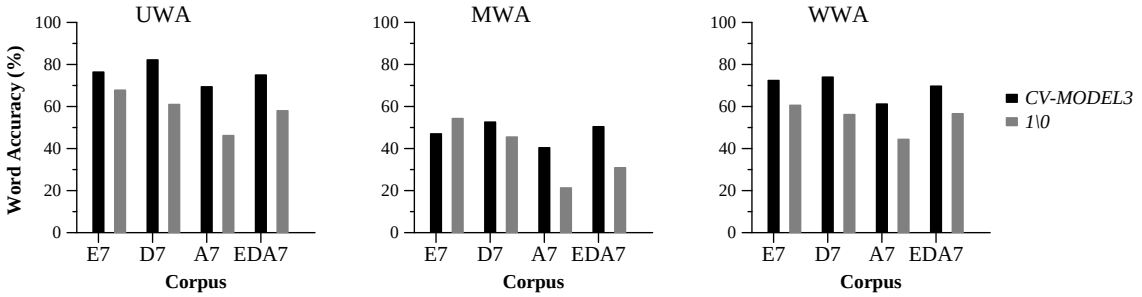


Figure 6.4: Comparison of the three evaluation metrics using two systems on four corpora.

all corpora for all metrics except MWA on E_7. This implies that ranking system performance on the same corpus most likely yields a system ranking that is transferable to other corpora. To further investigate, we randomly extracted 100 corpora of 500 word pairs from EDA_7 and ran the two systems on them and evaluated the results using UWA, MWA and WWA. All of the measures ranked the systems consistently using all these corpora (Figure 6.5).

As expected, the UWA metric is consistently higher than the MWA metric; it allows for the top transliteration to appear in any of the possible variants for that word in the corpus, unlike the MWA metric which insists upon a single target word. WWA falls in between the two other metrics; it respects all the transliteration variants but it values them based on the number of times they are suggested by transliterators. For example, for the E_7 corpus using the CV-MODEL3 approach, UWA is 76.4%, WWA is 72.4% and MWA is 47.0%.

6.4 Effect of Transliterator

The effect of individual transliterators on transliteration accuracy is studied in this section. Transliterator's provided transliterations are examined against the two systems and evaluated using word accuracy metric. Detailed results, transliterator separated, are shown in Table 6.2 and Table 6.3 for 1\0 and CV-MODEL3 respectively. Figure 6.6 summarizes these two tables on TOP-1 results. Noticeable difference in the measured accuracy of the systems using just one transliterators suggested target word makes the judgment among systems performances based on such corpora misleading. For example, evaluating CV-MODEL3 using EDA_7 made from H_1 suggestions yields 28.5% TOP-1 accuracy, where the same system is 50.5% accurate in its TOP-1 results for H_3, a 22.0% absolute difference. Despite the notable difference of

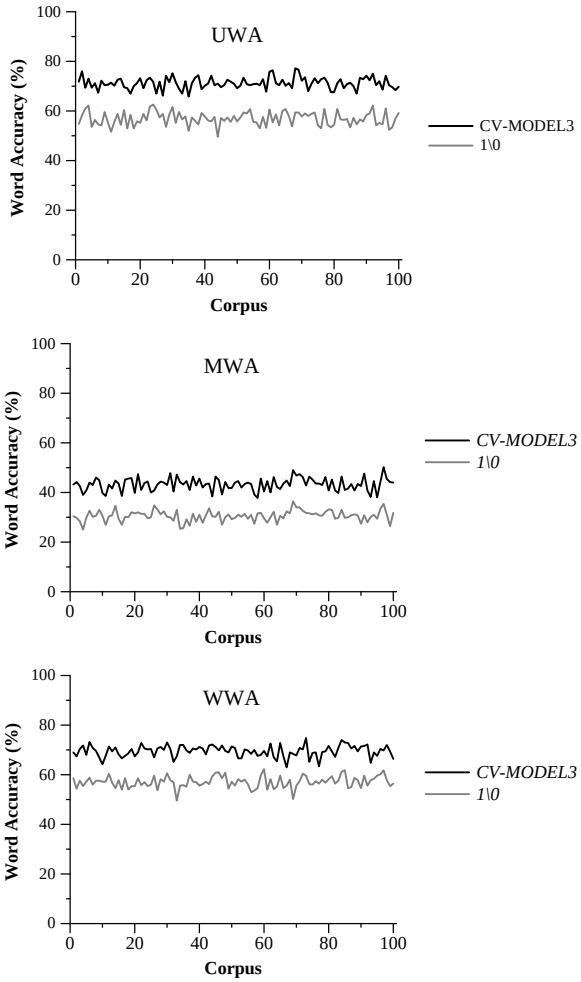


Figure 6.5: Comparison of the three evaluation metrics using the two systems on 100 random corpora. Note, data is discrete, but plotted as lines for clarity.

Transliterator		E_7	D_7	A_7	EDA_7
H_1	TOP-1	32.2	17.2	18.8	22.0
	TOP-5	58.4	39.0	53.0	49.3
	TOP-10	64.2	47.8	60.6	56.5
H_2	TOP-1	37.8	31.8	20.4	30.9
	TOP-5	67.8	60.8	59.2	63.3
	TOP-10	74.0	66.4	72.2	70.9
H_3	TOP-1	42.6	27.0	32.8	33.5
	TOP-5	69.2	44.8	73.0	63.7
	TOP-10	72.8	48.8	80.2	69.0
H_4	TOP-1	35.0	38.6	22.2	32.0
	TOP-5	63.6	62.4	57.4	62.1
	TOP-10	72.4	67.2	65.6	70.0
H_5	TOP-1	47.8	37.6	24.0	36.2
	TOP-5	72.4	60.8	61.6	67.3
	TOP-10	76.0	65.4	68.6	74.5
H_6	TOP-1	35.6	17.4	21.4	19.8
	TOP-5	61.0	43.2	59.4	52.3
	TOP-10	69.8	50.2	68.2	62.1
H_7	TOP-1	37.0	39.0	21.2	34.3
	TOP-5	65.2	56.0	67.0	63.0
	TOP-10	71.2	60.4	74.8	69.2

Table 6.2: Mean word accuracy using the $1\backslash 0$ *system on four corpora constructed separately from transliterations provided by seven transliterators.*

Transliterator		E_7	D_7	A_7	EDA_7
H_1	TOP-1	37.2	23.2	31.0	28.5
	TOP-5	61.2	50.6	61.2	58.6
	TOP-10	66.6	57.4	67.4	65.4
H_2	TOP-1	43.0	46.6	34.0	40.2
	TOP-5	76.2	66.4	71.6	71.3
	TOP-10	82.6	71.6	78.0	77.3
H_3	TOP-1	45.4	56.2	38.6	50.5
	TOP-5	74.2	78.8	79.6	79.1
	TOP-10	80.4	84.0	84.4	85.0
H_4	TOP-1	43.0	42.8	34.4	38.2
	TOP-5	74.0	66.8	69.8	70.6
	TOP-10	77.0	72.0	76.8	76.9
H_5	TOP-1	50.0	54.4	42.0	46.9
	TOP-5	76.4	75.2	72.6	74.9
	TOP-10	80.6	77.4	77.4	80.9
H_6	TOP-1	38.2	26.6	32.4	28.9
	TOP-5	69.4	52.0	69.0	59.8
	TOP-10	76.8	59.2	76.4	69.2
H_7	TOP-1	40.0	52.2	37.4	42.4
	TOP-5	72.4	65.8	74.0	72.6
	TOP-10	76.8	68.4	81.0	77.1

Table 6.3: Mean word accuracy using the CV-MODEL3 *system on four corpora constructed separately from transliterations provided by seven transliterators.*

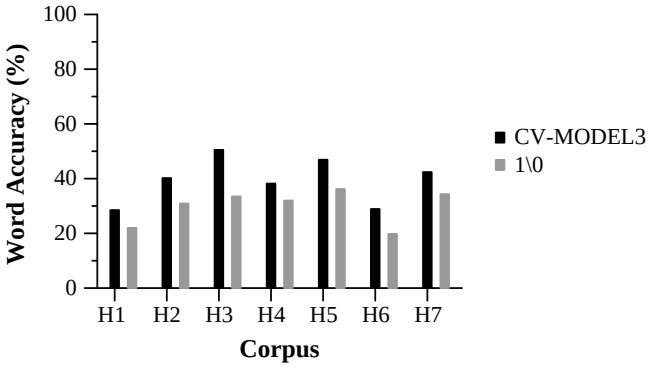


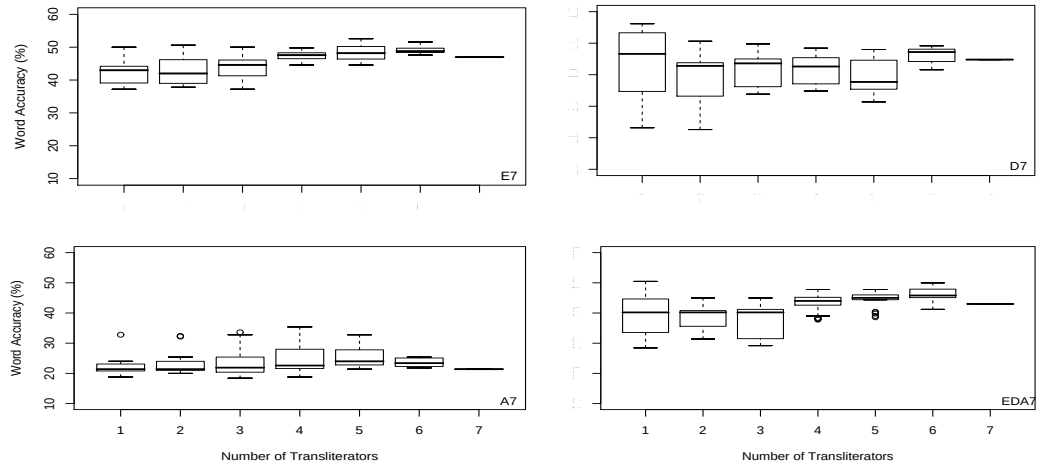
Figure 6.6: Mean word accuracy (TOP-1) *using two systems on corpora constructed separately from transliterations provided by seven transliterators.*

actual numbers, the ranking of the systems are still stable across the different transliterators' corpora.

We repeat the previous experiment using all the possible combinations of the available seven transliterators. In other words, each of the three sub-corpora, E_7, D_7, and A_7, and also the entire EDA_7 corpus is further divided based on the seven individual transliterators in different combinations. That is, we construct a sub-corpora from H_1's transliterations, H_2's, and so on; then take all combinations of two transliterators, then three, and so on. In general we can construct ${}^7C_r$ such corpora from r transliterators in this fashion, all of which have 500 source words, but may have anywhere between one to seven different transliterations for each of those words. We evaluated all these combinations using the three metrics. Detailed results of MWA are listed in Table 6.4 for CV-MODEL3 and Table 6.5 for 1\0. These results are also shown in Figure 6.7 which shows the MWA (TOP-1) for these sub-corpora. The x-axis shows the number of transliterators used to form the sub-corpora. For example, when $x = 3$, the performance figures plotted are achieved on corpora when taking all triples of the seven transliterator's transliterations.

From the boxplots it can be seen that performance varies considerably when the number of transliterators used to determine a majority vote is varied. However, the changes in performance as the x-axis is varied do not follow a fixed trend across the languages. For E_7, the range of accuracy achieved is high when only two or three transliterators are involved, ranging from 37.0% to 50.6% in CV-MODEL3 method and from 33.8% to 48.0% in 1\0 when

CV-MODEL3



1\0

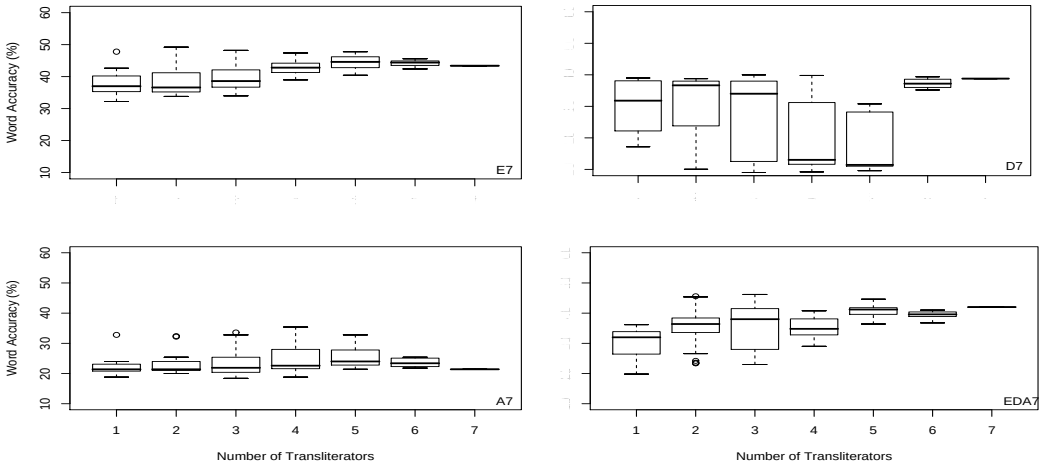


Figure 6.7: Performance on sub-corpora derived by combining the number of transliterators shown on the x-axis. Boxes show the 25^{th} and 75^{th} percentile of the MWA *for all ${}^{7}C_{x}$ combinations of transliterations, with whiskers showing extreme values.*

		E_7	D_7	A_7	EDA_7
${}^7C_1$	TOP-1	49.2	43.1	35.7	39.4
	TOP-5	77.5	65.1	71.1	69.6
	TOP-10	81.2	70.0	77.3	76.0
${}^7C_2$	TOP-1	43.0	43.5	36.6	44.7
	TOP-5	73.3	62.2	72.1	75.2
	TOP-10	78.5	66.7	78.6	80.6
${}^7C_3$	TOP-1	44.2	40.2	37.1	43.6
	TOP-5	74.9	57.4	73.4	73.4
	TOP-10	80.2	61.0	79.7	78.7
${}^7C_4$	TOP-1	47.5	30.7	36.5	45.4
	TOP-5	76.2	44.2	72.5	75.4
	TOP-10	81.5	47.4	79.2	80.6
${}^7C_5$	TOP-1	48.3	25.0	37.3	51.1
	TOP-5	75.0	38.8	74.0	80.3
	TOP-10	79.1	42.3	80.1	84.5
${}^7C_6$	TOP-1	49.2	51.7	39.7	53.7
	TOP-5	77.5	72.1	74.6	79.4
	TOP-10	81.2	74.9	81.8	82.7
${}^7C_7$	TOP-1	47.0	52.6	40.4	50.4
	TOP-5	76.8	74.6	74.2	81.6
	TOP-10	81.0	77.6	81.4	85.4

*Table 6.4: Mean word accuracy (*MWA*) for different subjects combinations using* CV-MODEL3 *system.*

only two transliterators data are available. When more than three transliterators are used, the range of performance is noticeably smaller. Hence it seems that if at least four transliterators

		E_7	D_7	A_7	EDA_7
${}^7C_1$	TOP-1	38.3	29.8	23.0	29.8
	TOP-5	65.4	52.4	61.5	60.1
	TOP-10	71.5	58.0	70.0	67.4
${}^7C_2$	TOP-1	38.9	31.1	22.7	35.2
	TOP-5	66.0	51.6	63.6	64.7
	TOP-10	71.9	56.3	71.3	71.0
${}^7C_3$	TOP-1	39.7	28.4	24.1	35.3
	TOP-5	67.5	47.3	64.8	64.2
	TOP-10	72.9	51.6	72.9	70.7
${}^7C_4$	TOP-1	42.8	20.7	24.6	35.1
	TOP-5	69.1	35.9	65.3	64.3
	TOP-10	74.0	40.0	73.8	69.7
${}^7C_5$	TOP-1	44.4	16.5	25.4	40.6
	TOP-5	69.6	30.8	65.1	69.9
	TOP-10	74.7	34.6	73.9	75.2
${}^7C_6$	TOP-1	44.2	37.2	23.6	39.4
	TOP-5	70.6	62.5	66.4	69.6
	TOP-10	75.4	68.2	74.8	75.3
${}^7C_7$	TOP-1	43.4	38.8	21.4	42.0
	TOP-5	69.4	65.2	64.6	73.0
	TOP-10	73.2	71.6	73.6	77.6

*Table 6.5: Mean majority word accuracy (*MWA*) for different subjects combinations using* $1\backslash 0$ *system.*

are used, then there is more chance that the MWA will be similar to an MWA calculated on a corpus derived using four different human transliterators.

The corpora derived from A_7 show consistent median increases as the number of transliterators increases, but the median accuracy is lower than for other languages. The D_7 collection does not show any stable results until at least six transliterator's are used. For example, when only two transliterators are involved, we get a range of 10.0% to 38.8% using 1\0 method and 23.2% to 50.6% using CV-MODEL3. The median increases with the number of transliterators drops when we add the fifth transliterator's dataset to the collection.

The results indicate that creating a collection used for the evaluation of transliteration systems, based on a "gold standard" created by only one human transliterator may lead to word accuracy results that could be at least 10% absolute different to results on a corpus derived using a different transliterator. This is evidenced by the leftmost box in each panel of the figure which has a wide range of results.

The language of origin also has an effect: accuracy for the Arabic collection (A_7) is generally less than that of English (E_7). The Dutch collection (D_7), is unstable through transliterators. In other words, accuracy differs in a narrower range for Arabic and English, but in wider range for Dutch. This is likely due to the fact that most transliterators found Dutch a difficult language to work with, as reported in Table 2.5.

6.4.1 Analysis of Evaluation using Combined Corpora

The wide range of accuracies obtained using less than four transliterators suggests the requirement of constructing transliteration corpora using at least four human transliterators. In this section, we use error analysis and hypothesis testing to more closely examine the conclusion taken from the previous experiments. The variability of the means of system accuracies can be measured by standard error of the mean [Butler, 1985] with the general formula of

$$e = Z_{\alpha/2}\frac{\sigma}{\sqrt{h}}, \tag{6.1}$$

which means the calculated mean word accuracy ($\overline{A}$) is estimated to vary in a range of $\overline{A} \pm e$, where σ is standard deviation of the accuracies, and $Z_{\alpha/2}$ is 1.96 for normal distribution ($\alpha = 0.025$). We consider h in Equation 6.1 as number of transliterators involved in corpus construction. For the two systems shown in Figure 6.7 on page 155, standard error is cal-

1\0				CV-MODEL3			
h	$\overline{A}$	σ	e	h	$\overline{A}$	σ	e
1	–	–	–	1	–	–	–
2	36.9	9.2	12.8	2	49.2	10.4	14.4
3	45.3	10.1	8.9	5	61.6	10.7	9.4
4	50.3	7.9	9.0	3	63.9	8.3	9.4
5	53.3	9.0	8.8	4	68.0	8.8	8.6
6	59.5	6.8	5.4	6	72.4	6.8	5.4
7	60.8	7.2	5.3	7	79.6	4.7	3.5

Table 6.6: Standard error of the mean (e) for two systems with different number of transliterators (h). Evaluation is performed on EDA_7 *corpus using* UWA (TOP-1).

culated as reported in Table 6.6. When the number of transliterators, h, is increased, error e is decreased. Therefore, the average accuracy $\overline{A}$ obtained with a corpus made using more transliterators, is closer to its real value. For example if only two transliterators are used, mean accuracy is 36.9 ± 12.8 for 1\0 and 49.2 ± 14.4 for CV-MODEL3 making differentiating these two systems difficult. However, when seven transliterators are used these values are 60.8 ± 5.3 and 79.6 ± 3.5 respectively, allowing more confidence in concluding they are different.

We can also perform a hypothesis test for the difference of systems using two-sample pooled t-test on normal populations taken from the independent observations made. Standard deviations of the systems, σ_1 and σ_2, are unknown. We assume σ_1 belongs to 1\0 and σ_2 belongs to CV-MODEL3. The hypotheses are set as below:

$$\begin{aligned} \text{H}_0 : &\quad \mu_1 = \mu_2, \\ \text{H}_1 : &\quad \mu_2 > \mu_1. \end{aligned}$$

H_0 indicates that the mean values (μ_1 and μ_2) are taken from the same population, and H_1 indicates that they are different. When real standard deviations (σ_1 and σ_2) are

unknown, if $\mu_1 - \mu_2 = d_0$ and the number of samples are lower than 30 (we have at most 7 transliterators), then we define the *t-value* for this sample as

$$\mathcal{T} = \frac{\overline{A}_2 - \overline{A}_1 - d_0}{S_p\sqrt{(1/h_1) + (1/h_2)}},$$

where

$$S_p = \sqrt{\frac{(h_1 - 1)S_1^2 + (h_2 - 1)S_2^2}{h_1 + h_2 - 2}},$$

where S_p is a pooled estimate of the common population standard deviation, with S_1^2 and S_2^2 representing the samples standard deviations, and $\overline{A}_1$ and $\overline{A}_2$ are the samples means. The degree of freedom is $v = h_1 + h_2 - 2$. In our problem $d_0 = 0$ and $h_1 = h_2$, therefore

$$\mathcal{T} = \frac{\overline{A}_2 - \overline{A}_1}{S_p\sqrt{2/h_1}},$$

$$S_p = \sqrt{\frac{S_1^2 + S_2^2}{2}},$$

$$v = 2(h_1 - 1).$$

We can test whether the systems are different, with CV-MODEL3 being superior over 1\0 (H_1), or there is no difference between them (H_0). The results are given in Table 6.7 (means and standard deviations are given in Table 6.6). According to this test, using EDA_7 corpus for evaluation if less than five transliterators are used, H_0 cannot be rejected, and therefore we cannot claim that CV-MODEL3 outperforms 1\0.

6.4.2 Transliterator Consistency

To further investigate why the perceived difficulty of transliterating a set of words might translate into reduced accuracy of a machine transliteration system that operates on such a derived corpus, we extracted further details of each of the E_7, A_7 and D_7 sub-corpora for each transliterator. Table 6.8 shows mean the number of distinct Persian characters used by each transliterator on each sub-corpus; and the average number of transformation rules generated by CV-MODEL3 on the ten training sets derived in the ten-fold cross validation process. For example, when transliterating words from E_7 into Persian, H_3 only ever used 21 out of 32 characters available in the Persian alphabet; H_7, on the other hand, used 24 different

h	S_p	$\mathcal{T}$	ν	t	$\mathcal{T} > t_\alpha$
2	9.8	1.255	2	4.303	–
3	10.4	1.919	4	2.776	–
4	8.1	2.374	6	2.447	–
5	8.9	2.611	8	2.306	H_0 rejected
6	6.8	3.286	10	2.228	H_0 rejected
7	6.1	5.765	12	2.179	H_0 rejected

Table 6.7: Hypothesis test for the difference of the two transliteration systems evaluated over EDA_7 *corpus using* UWA *metric* (TOP-1) *with* $\alpha = 0.025$.

	E_7		D_7		A_7		EDA_7	
	Char	Rules	Char	Rules	Char	Rules	Char	Rules
H_1	23	523	23	623	28	330	31	1075
H_2	22	487	25	550	29	304	32	956
H_3	21	466	20	500	28	280	31	870
H_4	23	497	22	524	28	307	30	956
H_5	21	492	22	508	28	296	29	896
H_6	24	493	21	563	25	313	29	968
H_7	24	495	21	529	28	299	30	952
Mean	23	493	22	542	28	304	30	953

Table 6.8: Number of distinct characters used and transformation rules generated per transliterator using CV-MODEL3.

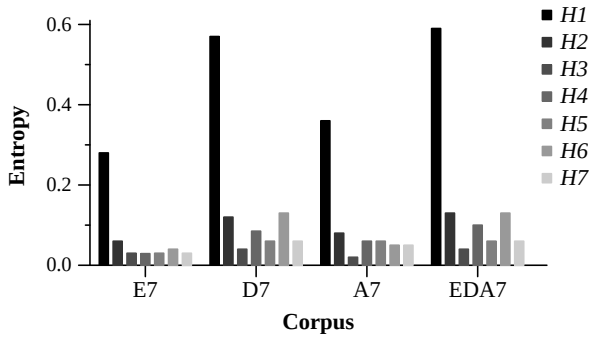


Figure 6.8: Entropy of the generated segments based on the corpora created by different transliterators.

Persian characters. It is expected that an increased number of characters or rules provides more "noise" for the automated system, hence may lead to lower accuracy. Superficially the opposite seems true for rules: the mean number of rules generated by CV-MODEL3 is much higher for the EDA_7 corpus than for the A_7 corpus, and yet Figure 6.4 shows that word accuracy is higher on the EDA_7 corpus. A correlation test, however, reveals that there is no significant relationship between either the number of characters used, nor the number of rules generated, and the resulting word accuracy of CV-MODEL3 (Spearman correlation, $p = 0.09$ (characters) and $p = 0.98$ (rules)).

What may give a better indication of "noise" in the corpus is the consistency with which a transliterator applies a certain rule. For example, a large number of rules generated from a particular transliterator's corpus may not be problematic if many of the rules get applied with a low probability. If, on the other hand, there were many rules with approximately equal probabilities, the system may have difficulty distinguishing when to apply some rules, and not others. One way to quantify this affect is to compute the *self entropy* of the rule distribution for each segment in the corpus for an individual. If p_{ij} is the probability of applying rule $1 \leq j \leq m$ when confronted with source segment i, then

$$D_i = -\sum_{j=1}^{m} p_{ij} \log_2 p_{ij},$$

is the entropy of the probability distribution for that rule. D is maximized when the probabilities p_{ij} are all equal, and minimized when the probabilities are very skew [Shannon, 1948]. As an example, consider the rules: $t \rightarrow$ (ت,0.5), $t \rightarrow$ (ط,0.3) and $t \rightarrow$ (د,0.2); for

which $D_{\mathrm{t}} = 0.79$.

In order to compute a single entropy value over the entire corpus as transliterated by a single transliterator, we could simply sum the entropies of the individual rule distributions. This assumes that each segment (the head of a rule) appears in the corpus the same number of times. A more accurate measure would be to compute the expected entropy

$$E = -\sum_{i=1}^{R} \frac{f_i}{F} D_i,$$

where D_i is entropy of the rule probabilities for segment i, R is the total number of segments, f_i is the frequency with which segment i occurs at any position in all source words in the corpus, and F is the sum of all the f_i's.

The expected entropy for each transliterator is shown in Figure 6.8, separated by corpus. Comparison of this graph with Figure 6.6 shows that generally transliterators that have used rules inconsistently generate a corpus that leads to low accuracy for the systems. For example, H_1 who has the lowest accuracy for all the collections in both methods, also has the highest expected entropy of rules for all the collections. For the E_7 collection, the maximum accuracy of 50.0%, belongs to H_5 who has the minimum expected entropy. The same applies to the D_7 collection, where the maximum accuracy of 56.2% and the minimum expected entropy both belong to H_3. These observations are confirmed by a statistically significant Spearman correlation between expected rule entropy and word accuracy ($r = -0.54, p = 0.003$). Therefore, the consistency with which transliterators employ their own internal rules in developing a corpus has a direct effect on system performance measures.

6.4.3 Inter-Transliterator Agreement and Perceived Difficulty

Here we present various agreement proportions, P_G from Equation 2.16 on page 64, which give a measure of consistency in the corpora across all users, as opposed to the entropy measure which gives a consistency measure for a single user. For E_7, P_G was 33.6%, for A_7 it was 33.3% and for D_7, agreement was 15.5%. In general, humans agree less than 33% of the time when transliterating English to Persian.

In addition, we examined agreement among transliterators based on their perception of the task difficulty shown in Table 2.5. For A_7, agreement among those who found the task *easy* was higher (22.3%) than those who found it in *medium* level (18.8%). P_G is 12.0% for

Metric	Perception	1\0	CV-MODEL3	Relative Improvement
UWA	Easy	33.4	55.4	54.4 ($p < 0.0001$)
	Medium	44.6	48.4	8.52 ($p < 0.0001$)
MWA	Easy	23.2	36.2	56.0 ($p < 0.0001$)
	Medium	30.6	37.4	22.2 ($p = 0.0380$)
WWA	Easy	35.4	52.2	47.4 ($p < 0.0001$)
	Medium	22.8	33.0	44.7 ($p < 0.0001$)

Table 6.9: System performance (in percent) when A_7 is split into sub-corpora based on transliterators perception of the task (Easy or Medium).

those who found the D_7 collection *hard* to transliterate; while the six transliterators who found the E_7 collection difficulty *medium* had $P_G = 30.2\%$. Hence, the harder participants rated the transliteration task, the lower the agreement scores tend to be for the derived corpus.

Finally, in Table 6.9 we show word accuracy results for the two systems on corpora derived from transliterators grouped by perceived level of difficulty on A_7. It is readily apparent that CV-MODEL3 outperforms on the corpus comprised of human transliterations from people who saw the task as easy with both word accuracy metrics; the relative improvement of over 50% is statistically significant ($p < 0.0001$, paired t-test on ten-fold cross validation runs). However, on the corpus composed of transliterations that were perceived as more difficult, "Medium", the advantage of CV-MODEL3 is significantly eroded, but is still statistically significant for UWA and WWA. Here again, using only one transliteration, MWA did not distinguish the performance of each system.

6.5 Conclusions

Along with the experimental results that compared different methods in the previous chapters (Chapter 3 to 5), we noticed the differences of performances of systems evaluated on different corpora, yet with the same source and target languages. To investigate what causes the

performances to differ, we conducted experiments on source language and corpus size factors on the E_7 and corpora. Our initial experiments revealed that language origin of source words that corpus is composed of, affects the accuracy of transliteration systems.

We have evaluated two English to Persian transliteration systems on a variety of controlled corpora using our defined evaluation metrics that modified the metrics that appear in previous transliteration studies. Varying the evaluation corpus in a controlled fashion has revealed several interesting facts.

We report that human agreement on the English to Persian transliteration task is about 33%. The effect that this level of disagreement on the evaluation of systems has, can be seen in Figure 6.6 (page 154), where word accuracy is computed on corpora derived from single transliterators. Accuracy can vary by up to 30% in absolute terms depending on the transliterator chosen. To our knowledge, human agreement, and its effects on transliteration accuracy has not been studied before.

In order to alleviate some of these effects on the stability of word accuracy measures across corpora, we recommend that each word in the corpus be transliterated by at least five transliterators to construct a corpus. Figure 6.7 shows that constructing a corpus with four or more transliterators, the range of possible word accuracies achieved is less than that of using fewer transliterators. We thereby addressed the question on whether the number of valid transliteration variants included in a corpus affect the comparison of performance of multiple systems using transliteration metrics we introduced. These metrics valued the transliteration variants, and also the number of variants that multiple transliterators introduced to a corpus.

Some past studies do not use more than a single target word for every source word in the corpus [Bilac and Tanaka, 2005; Oh and Choi, 2006]. Our results indicate that it is unlikely that these results would translate onto a corpus other than the one used in these studies, except in rare cases where human transliterators are in 100% agreement for a given language pair.

We also addressed our research question on the effect that the origin of the source words provided to transliterators has on the perceived accuracy of the transliteration systems. Given the nature of the English language, an English corpus can contain English words from a variety of different origins. In this study we have used English words from an Arabic and Dutch origin to show that word accuracy of the systems can vary by up to 25% (in

absolute terms) depending on the origin of English words in the corpus, as demonstrated in Figure 6.4. Hence, we recommend careful selection of source words for corpus construction. If the script of the source words do not represent their origins (that is, words have already been transliterated), the corpus requires enough similar origin words to support and build the transliteration model required for that language; otherwise, those words act as noisy inputs to the system.

In addition to computing agreement, we also investigated the transliterator's perception of difficulty of the transliteration task with the ensuing word accuracy of the systems. Interestingly, when using corpora built from transliterators that perceive the task to be easy, there is a large difference in the word accuracy between the two systems, but on corpora built from transliterators who perceive the task to be more difficult, the gap between the systems narrows. Hence, a corpus applied for evaluation of transliteration should either be made carefully with transliterators with a variety of backgrounds, or should be large enough and be gathered from various sources so as to simulate different expectations of its expected non-homogeneous users.

The self entropy of rule probability distributions derived by the automated transliteration system can be used to measure the consistency with which individual transliterators apply their own rules in constructing a corpus. It was demonstrated that when systems are evaluated on corpora built by transliterators who are less consistent in their application of transliteration rules, word accuracy of the automated systems is reduced.

Given the large variations in system accuracy that are demonstrated by the varying corpora used in this study, we recommend that extreme care be taken when constructing corpora for evaluating transliteration systems. Studies should also give details of their corpora that would allow any of the effects observed in this study to be taken into account.

In the previous chapters, we evaluated our transliteration systems using two corpora: B_{e2p} and B^{+}_{e2p}. To clarify how the results and conclusions drawn previously conform to the findings in this chapter, we review their evaluation process. The details of the corpora used for evaluation, their construction process and the origins of the source words, are listed in Chapter 2. B_{e2p} is composed of only English origin source words, while the B^{+}_{e2p} source words have multiple origins. Although only one transliterator was responsible in transliterating each source word (and therefore no transliteration variant is available), 26 different transliterators

have been involved in the construction of our relatively large corpora of 16,670 word-pairs. Hence, the transliteration habit of many transliterators is captured in the corpora, and should be representative of their society. Table 7.1 in the next chapter, presents our evaluation results of all the systems introduced in this thesis on EDA_7 to confirm the results previously stated.

Chapter 7

Conclusion

"One never notices what has been done; one can only see what remains to be done."

— Marie Curie

Machine transliteration systems input natural language words (mostly proper names and technical terms) and output their phonetical translation in another language. Automatic transliteration is challenging; it is sometimes even hard for humans in the absence of source or target language knowledge. The essential knowledge that transliteration demands is conventions of spellings and sounds in each of the languages, and inter-lingual transformation habits. An ideal generative transliteration system captures all of this information through a training stage, and then in practice it is able to apply to correct rules in the correct context to generate acceptable transliterations of a given source word. Practically however, existing transliteration systems are still far away from the ideal state for many languages, despite much research on phonetical, spelling-based, hybrid, or combined methods.

In this thesis, we explored the machine transliteration problem for English and Persian. This language-pair has not been explored before in any automatic transliteration study. All the approaches we investigated here were in the framework of spelling-based generative transliteration, to avoid the errors that the extra transformation steps of phonetic-based methods may introduce into the transliteration process. Also, spelling-based approaches are generally considered to be superior to phonetic-based methods because they do not rely on pronunciation dictionaries which may not include the pronunciations of all words. Overall,

we explored two aspects of the transliteration problem: transliteration effectiveness and evaluation, as summarised below.

7.1 Effective Machine Transliteration

A major part of this thesis was dedicated to transliteration algorithms, covering both the training and testing stages, all aiming to increase transliteration accuracy in the top-ranked positions of the suggested transliteration lists. To improve transliteration effectiveness, the following contributions have been made:

- Thorough examination of the effects of source and target language models on transliteration effectiveness;
- Novel approaches for both English-Persian and Persian-English transliteration;
- A novel algorithm for English-Persian alignment;
- A novel back-transliteration algorithm for restoring English words from their Persian transliterated word;
- Output combination of multiple transliteration systems in a black-box scheme.

Machine transliteration approaches investigated in this thesis can be divided into four categories (as partially shown in Table 7.1 evaluating on the EDA_7 corpus using UWA on the TOP-1 basis). The first category is a manual method which uses handcrafted rules listed in Appendix D. Such a system was used as a baseline for comparing the automatic systems, which learn their transliteration model through training, with a system that applies only a simple pre-defined model. Probabilities of the transformation rules are learnt from a bilingual transliteration corpus. The second category, called n-gram based, uses a fixed sized segmentation approach in both the training and transliteration steps. The third category represents the transliteration algorithms developed in this thesis, which approach the problem by defining the patterns of segmentation using consonants and vowels in the spellings of the words. These methods attempt to capture the right context for forming the transliteration models, rather than phonemes, graphemes, or fixed sized segments. The fourth category is a

Category	Method	English-Persian	Persian-English
Manual	Handcrafted rules	56.2 (3.2)	16.6 (1.3)
N-gram	1\0	59.7 (2.7)	13.6 (3.8)
(Chapter 3)	1\1	56.6 (3.5)	41.4 (2.6)
	2\2	57.4 (3.6)	42.2 (2.5)
	2\2T	48.3 (4.1)	47.1 (2.4)
Consonant-Vowel	CV-MODEL1	59.9 (3.9)	52.8 (3.3)
(Chapter 4)	CV-MODEL2	61.7 (3.1)	36.5 (2.2)
	CV-MODEL3	74.3 (3.8)	39.0 (2.6)
Combined	4-systems	85.5 (2.0)	69.5 (3.0)
(Chapter 5)			

Table 7.1: An overview of the performance of the best transliteration systems investigated in this thesis (measured using uniform word accuracy in TOP-1*) for English-Persian and Persian-English on the* EDA_7 *and* B_{p2e} *corpora, respectively. Standard deviations are given in parentheses.*

combination of the outputs of the previous systems, and employs majority voting and Naïve-Bayes classifier. According to this categorisation, and experiments reported in the previous chapters, our major findings are as follows.

From the results summarised in Table 7.1, and the evaluation results reported in the previous chapters, we can perceive the language-specific nature of transliteration: even swapping the source and target languages causes differences in the performance of the systems. For example, the best approach based on n-grams for English-Persian transliteration is 1\0 that uses only one past symbol as context. Persian-English transliteration, on the other hand, is worst with 1\0. More past and future context was required to generate more accurate outcomes. Similarly, using consonant-vowel patterns in transliteration did not produce identical results when the source and target languages were swapped. CV-MODEL3, which collapses runs of consonants and vowels, worked best for English-Persian, improving the transliteration accuracy by relative 24.4% over the best n-gram based approach. Persian-English translit-

eration, on the other hand, was best using the CV-MODEL1 approach improving the word accuracy by 12.1%. These results also suggest that Persian-English transliteration is more difficult than the reverse. This is largely due to lack of written short vowels in Persian script.

The most stable and effective method for transliteration of English and Persian, regardless of the direction, was a combination of the outputs of the most accurate individual systems: CV-MODEL1, CV-MODEL3, 1\0, 2\2. Improvements achieved in the effectiveness of transliteration were 15.1% and 29.7% for English-Persian and Persian-English, respectively. Other than performance improvement, we discovered a unique combination that worked well for both transliteration directions and reduced the gap of efficacy between them.

7.2 Transliteration Evaluation

Evaluation of transliteration systems has largely been overlooked in the literature. Most past research was solely focused on increasing the transliteration accuracy in terms of word accuracy or character accuracy, while the question of whether these results are comparable to the ones drawn from other corpora (same language-pair) was not addressed. The specification of the corpus that makes such judgments valid was also missed in these studies. Therefore, in the area of transliteration evaluation there were two main problems related to evaluation corpus and evaluation metric which we addressed in this thesis with the following contributions:

- Construction of bilingual transliteration corpora for English and Persian language-pair;
- Determining the effects of the corpus construction process on evaluation results, in terms of corpus size, language of origin of source words, and the number of transliterators.

The most important findings from our experiments can be summarised as follows.

A large bilingual transliteration corpus is desirable for transliteration studies; this provides more training data, which is useful in forming a more accurate transliteration model. However, our experiments on the corpora we constructed showed that if the origin of the source words is strictly controlled to disallow divergence, even a small corpus is enough for obtaining stable results from transliteration systems.

When transliteration evaluation is performed on a corpus where its construction procedure — particularly the number of transliterators — is unknown, the word or character accuracies are not reliable. They may change significantly when using another corpus, and are not even directly comparable to other studies reporting performance of other transliteration systems. Careful reporting of the corpus construction processes was therefore found to be essential.

A corpus that is constructed using more than three transliterators per source word reduces the errors that can arise in evaluations, possibly leading to incorrect judgments of the relative systems performance. Foreign language knowledge, and personal habits of transliteration, can cause disagreements between the human transliterators. While humans do not agree on transliteration of some words, systems are to provide coverage for that diversity by suggesting multiple transliterations (as a ranked list, with the highest ranked transliteration — TOP-1 being the item with most agreement). Such phenomena should be reflected both in the evaluation of systems and transliteration corpora. For example, if a source word S has two valid transliterations T_1 and T_2, and a transliteration system generates T_1, and then is evaluated on a corpus containing (S,T_2), it will be unfairly disadvantaged.

7.3 Future Work

We addressed the problem of English-Persian and Persian-English machine transliteration by developing new algorithms of segmentation and alignment. Other possible methods of transliteration remain to be investigated, which may improve the transliteration performance over what we reported here. Any advancement in the methods of detecting graphemes and phonemes in each language (source or target), similar to the methods under investigation for speech recognition, may be applicable for transliteration as well. There is also room for improvements in the alignment step if better detection of transliteration units (grapheme or phoneme) becomes available, especially through connecting these units in the source and target words.

System combination was shown to be beneficial for increasing the robustness of the transliteration systems, and also improved the overall transliteration accuracy. The main reason was that the errors of each system are not propagated to the others; particular weaknesses of one system can be covered by other systems, while their strengths were accumulated to the integrated system. There are many different techniques of system hybridisation and

combination reported in the literature of other natural language processing systems; these could be adopted for machine transliteration, and may also have positive effects on their performance. For example, other classification algorithms can be applied and compared to the Naïve-Bayes classifier we reported in our experiments.

One deficiency of the evaluations reported in this thesis is ignoring multi-word transliteration. Mostly applicable for Arabic, some names are composed of multiple parts and are transliterated to only one word in the target language, and vice versa. For example, one possible transliteration of the person name "عبدالحمید" (one word) can be "Abd al-hamid" (two words, or three words without hyphen). Some studies are reported to address this phenomenon using phrase-based machine translation techniques; however, we did not examine the effectiveness our methods on phrases.

Although we only investigated a particular language-pair (English and Persian), the basic ideas and algorithms can be modified for other languages. An ideal approach that could even be reflected in our intended language-pair would be an algorithm that distinguishes the patterns of segmentation (similar to CVC, CV, and VC in English-Persian transliteration) automatically from a training corpus. That way, any pair of languages could be transliterated under a unique framework.

In this thesis, we only initiated studies on the effects of corpus construction on transliteration evaluation. Despite the conclusions drawn from our experiments, the effort of making a corpus using more than one transliterator might discourage researchers from pursuing our recommendations of using four or more transliterators per word. That effort could be alleviated with the minimum number of source-target word-pairs that a corpus needs. For English-Persian only 500 pairs were shown to be sufficient. Another option is to use a large corpus that is built using multi-transliterators, with each person transliterating one part of the corpus. Care should be taken to employ transliterators with diverse backgrounds and knowledge. For studies of transliteration effectiveness to be comparable, and their percentages of accuracy to be trustable, they should provide details of the bilingual transliteration corpora they use. The same applies for initiatives to provide publicly available corpora for machine transliteration. In this thesis the effect of number of transliterations generated by systems on the evaluations is not reported which can be followed in further studies.

Appendix A

Abbreviations

List of all the abbreviations and symbols used throughout this text.

Corpus

B	a bilingual corpus
B_{e2p}	English to Persian corpus (English origin only) of 1,857 entries.
B^{+}_{e2p}	English to Persian corpus (multiple language origin) of 16,761 entries.
B_{p2e}	Persian to English corpus of 2,010 entries.
A_7	English to Persian corpus of 500 entries, Arabic origin, a subset of EDA_7.
D_7	English to Persian corpus of 500 entries, Dutch origin, a subset of EDA_7.
E_7	English to Persian corpus of 500 entries, English origin, a subset of EDA_7.
EDA_7	English to Persian corpus of 1500 entries with three origins of English, Dutch, and Arabic.

Accronyms

IPA	international phonetic alphabet
LDC	linguistic data consortium
MT	machine translation
SMT	statistical machine translation
NLP	natural language processing
OCR	optical character recognition

Variables

H	human transliterator
L	output list of a transliteration system M
M	transliteration system
S	source word (in transliteration), source sentence (in translation)
T	target word or a set of target words associated to one source word (in transliteration), target sentence (in translation)
T	indicate use of target language model in probability calculations
TOP-N	N highest probable transliterations for a given source word

C	Consonant
$\mathcal{C}$	Collapsed consonants
V	Vowel
$\mathcal{V}$	Collapsed vowels

Metrics

A	word accuracy
CA	character accuracy
MWA	majority word accuracy
UWA	uniform word accuracy
WWA	weighted word accuracy

Appendix B

Terminology

Throughout this thesis, we use some general terms from areas of linguistics, phonetics, and speech processing. A brief explanation of these terms is provided in this section[1].

Phonetics and Phonology

Phonetics is the study of the sounds of human speech, and is concerned with the actual properties of speech sounds, their production, audition and perception. Phonetics deals with the sounds independently rather than the contexts in which they are used in languages. Phonology, on the other hand, studies sound systems and abstract sound units, such as phonemes. In other words, phonetics are the same in all the languages; the difference is provided by phonology. The phonetic representation of a sound is shown using [ ], and the phonemes are represented by / /. For example, the phonetic version of the both Persian letter "پ", and the English letter "p" is [p].

Phoneme

A phoneme is the smallest unit of speech that distinguishes meaning. Phonemes are the important units of each word, and substituting them causes the meaning of a word to change. For example, if we substitute the sound [b] with [p] in the word "big" [bɪg], the word changes to "pig". Therefore /b/ is a phoneme.

[1] Definitions are taken from: Crystal [2006] and Wikipedia (http://en.wikipedia.org/).

Grapheme

A grapheme is the fundamental unit in written language. It includes alphabetic letters, Chinese characters, numerals, punctuation marks, and all the individual symbols of any writing system. In a phonemic orthography, a grapheme corresponds to one phoneme. In spelling systems that are non-phonemic — such as the spellings used most widely for written English — multiple graphemes may represent a single phoneme. These are called digraphs (two graphemes for a single phoneme) and trigraphs (three graphemes). For example, the word "ship" contains four graphemes (s, h, i, and p) but only three phonemes, because "sh" is a digraph.

Writing system

A writing system is a symbolic system for representing expressible elements or statements in language. A writing system has four sets of specifications:

1. a set of defined symbols that are individually called characters or graphemes, and collectively called a script;
2. a set of rules and conventions which arbitrarily assign meaning to the graphemes, their ordering, and relations, and are understood and shared by a community;
3. a language, whose its constructions are represented and recalled by the interpretation of these elements and rules; and
4. some physical means of distinctly representing the symbols by application to a permanent or semi-permanent medium, so that they may be interpreted.

Alphabetic or segmental writing systems possess an alphabet which is a small set of letters or symbols that represents a phoneme of a spoken language. Arabic and Latin writing systems are segmental (Persian and English are two languages of these two writing systems, respectively).

Consonant

In articulatory phonetics, a consonant is a sound in spoken language that is characterised by complete or partial closure of the upper vocal tract that lies above the larynx. Consonant is

a Latin word which means “sounding with” or “sounding together”. That means consonants do not sound on their own, but occur only with a nearby vowel. Although this is correct for Latin, in some languages such as Nuxálk consonants may occur without any vowels.

Vowel

In phonetics, a vowel is a sound in spoken language that is characterised by an open configuration of the vocal tract so that there is no build-up of air pressure above the glottis. A vowel is known to be syllabic.

The semantic significance of vowels varies widely depending on the language. In some languages, particularly Semitic languages, vowels mostly serve to represent inflections. The alphabets used to write the Semitic languages, such as the Hebrew alphabet and the Arabic alphabet (used also in Persian), do not ordinarily mark all the vowels.

Consonant cluster

A consonant cluster is a group of consonants that have no intervening vowel. Some linguists limit the cluster to the boundaries of a syllable, but we do not input this limitation in this thesis.

Vowel sequence

A vowel sequence is a group of adjoining vowels only.

Monophthong

A monophthong is a pure or stable vowel sound whose articulation at both the beginning and end is relatively fixed, and does not glide up or down towards a new position of articulation. All languages have monophthongs and many languages have diphthongs.

Diphthong

In phonetics, a diphthong is a monosyllabic vowel combination involving a quick but smooth movement from one vowel to another. It is often interpreted by listeners as a single vowel sound or phoneme. In phonology, diphthongs are distinguished from sequences of monophthongs when the vowel sound is analysed into different phonemes.

Approximant

An approximant is a speech sound that is intermediate between vowels and consonants. For example, the sound [w] in *well* is an approximant.

Semivowel

A semivowel, also known as glide or semi-consonants, is an intermediate between vowel and consonant. Semivowels are a non-syllabic vowels that forms diphthongs with syllabic vowels. They may be contrasted with approximants which are closer to vowels, where semivowels behave more similarly to consonants.

Elision

Elision is the omission of one or more sounds (such as a vowel, a consonant, or a whole syllable) in a word or phrase, to make the word easier for the speaker to pronounce. Even though the pronunciation of a word should not influence writing, a word or phrase is sometimes spelled the same as it is spoken; for example, in poetry or in the script for a theatre play, in order to show the actual speech of a character. It may also be used in an attempt to transcribe non-standard speech, or in transliteration when some source sounds are missing in the target language.

Epenthesis

In informal speech epenthesis occurs within unfamiliar or complex consonant clusters. In linguistics, epenthesis generally breaks up a vowel sequence or a consonant cluster that is not permitted by the phonotactics of a language. An example of occurrence of epenthesis is in transliteration.

Allophone

In phonetics, an allophone is one of several similar phones that belong to the same phoneme. For example, some letters in Persian script are allophones; such as “س”, “ث”, and “ص” which are all pronounced as /s/.

Appendix C

International Phonetic Alphabet (IPA)

The international phonetic alphabet (IPA) is a system of phonetic notation based on the Latin alphabet. It a standard representation of the sounds of spoken language and is developed by the international phonetic association [IPA, 1999][1]. List of phonetic notations used throughout this thesis accompanied by an English example is listed below [2]:

IPA Sound Notation	English	Persian	English Example
[ɒ]	a	ا	f<u>a</u>ther
[æ]	a	—	c<u>a</u>t
[b]	b	ب	<u>b</u>ad
[s]	c, s	س, ص, ث	<u>s</u>ad
[tʃ]	ch	چ	<u>ch</u>at
[d]	d	د	<u>d</u>ad
[e],[ɛ]	e	—	p<u>e</u>t
[ɜ]	e(r),u(r),i(r)	—	b<u>i</u>rd
[f]	f	ف	<u>f</u>at

[1] http://en.wikipedia.org/wiki/International_Phonetic_Alphabet
[2] http://en.wikipedia.org/wiki/Persian_phonology

IPA Sound Notation	English	Persian	English Example
[g]	g	گ	get
[h]	h	ه, ح	hat
[dʒ]	j	ج	jet
[k]	c,k,q	ک	cat
[l]	l	ل	let
[m]	m	م	mat
[n]	n	ن	nut
[o]	o	—	more
[p]	p	پ	pet
[ɾ]	r	ر	rat
[s]	s	س, ص, ث	sat
[ʃ]	sh	ش	shed
[t]	t	ت, ط	ted
[θ]	th	—	thing
[uː]	u, oo, ou	و	soon
[v]	v	و	vet
[w]	w	و	wet
[ɯ]	wh	و	what
[x]	—	خ	—
[j]	y	ی	yet
[eɪ]	ei, ay, ai	ی	hey
[iː]	i, ee, y	ی	see
[z]	z	ز, ض, ظ, ذ	z
[ʒ]	—	ژ	pleasure
[ɣ]	—	ق, غ	—

Appendix D

Handcrafted English-Persian and Persian-English Transformation Rules

Manually made transformation rules that are used for both English-Persian and Persian-English transliteration (with a change of source and target) as a baseline, is listed below (probabilities can be calculated using a training bilingual corpus).

Persian	English	Persian	English	Persian	English
ا	a	خ	kh	ظ	z
ا	aa	د	d	ع	’
ب	b	ذ	z	ع	a
پ	p	ر	r	غ	q
ت	t	ز	z	غ	gh
ت	th	ژ	zh	ف	f
ث	s	ژ	j	ف	ph
ح	j	س	s	ق	q
ح	g	س	c	ق	gh
چ	c	ش	sh	ک	k

APPENDIX D. HANDCRAFTED ENGLISH-PERSIAN AND PERSIAN-ENGLISH TRANSFORMATION RULES

Persian	English	Persian	English	Persian	English
چ	ch	ص	s	ک	c
ح	h	ض	z	ک	ck
خ	x	ط	t	گ	g
ل	l	م	m	ن	n
و	v	و	u	و	o
ه	h	ی	y	ی	i
ی	a				

Appendix E

A Sample of English-Persian Transliteration Corpus

A selected sample of the EDA_7 corpus is given below:

English	Persian Variants
Abraham	ابراهام
Addison	ادیسن ،ادیسون
Abdalmalik	عبدالملک ، عبدالمالیک ،ابدلمالیک ، عبدلمالیک
Abeltje	ابلج ،ابلت ،ابلتج
Allington	الینگتن ،الینتن ،الینگتون ،النگتن
Ambrosius	امبروسوس ،امبروسیوس ،امبروشس
Annachet	انچت ،اناچت
Appleton	اپلیتن ،اپلتون
Ashley	اشلی
Axford	اکسفرد ،اکسفورد
Baefje	بایف ، بیفج ، بیف ، بفجه ، بالفج
Baldington	بلدینگتون ، بلدینگتن ، بالدینگتن ، بالدینگتون
Barhydt	بارهیدت ، بارهیت ، بریت ، بارهایت
Bauduyn	بادین ، بیودیون ، بادیون ، باودین
Basil	بصیل ، بسیل ، باسیل
Bewick	بیویک ، بویک

English	Persian Variants
Bosscha	بوسشا ، باسچا ، بوسچا ، بوسکا
Botros	بتروس ، بوستروس
Brigje	بریگیه، بریگجه ، بریژ
Brinkerhoff	برینکرهف ، برینکرهاف ، برینکرهوف
Calcutt	کالکوت ،کالکات ،کلکیوت ،کلکات
Christiaen	چریستین،کریستین
Christopher	کریستفر ، چریستوفر ،کریستوفر
Coenraedt	کوینراد ،کوینرات ،کونرایت ،کوینرایت
Crockford	کروکفورد ،کراکفرد ،کراکفورد
Dawson	داسن ، داوسن ، داوسون ، داوسان
Dennell	دننل ، دنل ، دینل
Dierderick	دیردرک ، دیردریک ، دایردریک
Dominicus	دمینیکوس ، دومینیکاس ، دومینیکوس ، دامینیکاس
Ebsworth	ابسورس ،ایبسورث ،ابسورث
Eduwart	ادووارت ،ادوارت
Emmanuel	امانوال ،امانول ،امانویل
Franklin	فرانکلین ، فرنکلین
Francytje	فرانسیتیه ،فرانسیت ،فرانسیتج ، فرنسایت ، فرانسیتژ
Gallaway	گالاوی ،گالوی
George	جرج ، جورج
Geertruyd	گرترویدگیرترویدگ
Gordon	جردن ،گردون ،گوردن ،گردن ،گوردون
Gorinchem	گورینچم ،گوروینچم ،گروینچم
Hampshire	هامپشیر ، همپشایر ، هامپشایر
Hazlewood	هزلیوود ، هازلوود ، هزلوود
Heemstede	هیمستد ، همستد ، هیمستده ، هیمستد
Huyge	هویج ، هویگ ، هایگ ، هایج
Iman	ایمان
Irving	ایروینگ،اروینگ
Isaac	ایزاک،ایساک
Jackman	جکمن
Jacob	ژکوب ، جکوب ، جاکوب

English	Persian Variants
Jamal	جمال ، جمل
Joseph	جسف ، جزف ، ژوزف ، جوزف
Kempster	کمپاستر ،کمستر ،کمپستر
Knapp	ناپ ، نپ
Koenraad	کوانراد ،کونراد ،کوینراد
Lambourne	لامبورنه ، لمبرن ، لامبورن ، لمبورن ، لمبوورن
Lambrecht	لمبرچت ، لامبرچت ، لامبرچ ، لمبرچ
Lawrence	لاورنس ، لورنس ، لارنس
Lockington	لاکینگتون
Machteld	مچلد ،مچتلد ، ماچتلد
Magdalen	ماگدلن ، ماگدالن ، مگدالن
Markham	مارخام، مارکهام
Michiel	میچل ، مایکل ، میشیل ، میشل
Moreby	مربی ، موربی ، موربای
Newington	نیواینگتن ، نیواینگتون ، نیووینگتون
Nichols	نیکلز ، نیکولس، نیکولز
Osterhoudt	اوسترهوت ،استرهات ،استرهوت
Oudedelft	اددلت ،اوددلت ،اوددلف ،اوددلفت
Pardington	پاردینگتن ، پاردینگتون
Paterson	پاترسون ، پترسون
Phillips	فیلیپس
Pieter	پایتر ، پیتر
Pritchard	ریچارد ، پریتچارد ، پریچارد
Qadisiyah	قدسیا ، قدسیه ،کدیسیه ، قادسیه
Quackenboss	کوکنبوس ،کواکنبوس ،کویکنباس ،کویکنبوس
Quinton	کیونتن ،کوینتن ،کویینتن ،کواینتون
Resolveert	ریسالورت ، رسولورت ریسولویرت ، ریزولورت ، ریسولوّرت
Richardson	ریچاردسن، ریچاردسون
Rowland	رولند ، رولاند
Ruudt	رادت ، روت ، ریودت
Russell	روسل ، راسل
Salmon	سالمن ، سالمون
Sasze	سازی ، سازه ، ساسز

English	Persian Variants
Schermerhorn	اسچرمرهورن ، اسکیرمیرهرن ، اسکرمرهورن ، شرمرهورن
Sheldon	شلدن ، شیلدون ، شلدون
Simmons	سیمنز ، سیمونز ، سیمونس
Talboys	تالبویس ، تلبویز ، تالبویز
Tamir	تمیر ، تامیر
Tenbrook	تنبروک
Thornton	تورنتن ترنتن ، تورنتون
Tomatius	تاماتیوس ، تومیتیوس ، توماتیوس
Trafford	ترفورد ، ترافورد
Utrecht	اترچت ، اترچ ، اوتریچت ، اوترچت
Upston	اپستون ، اپستون
Vancortlandt	وانکورتلند ، وانکورتلنت ، ونکورتلند ، وانکورتلندت ، ونکورتلندت
Verbeck	وربک
Voikins	وکینز ، ویکینس ، ویکینز ، وویکینز
Wallington	والینگتن ، والینگتون
Warland	وارلند
Wentworth	ونتوورث ، ونورت ، ونتورت ، ونتورث
Willemyn	ویلیمین، ویلمین
Wesselius	وسلوس ، وزلیوس ، وسلیوس
Yerbury	یربری
Yzaak	یزاک ، وایزاک ، زاک
Zammit	زمیت ، زامیت
Zoeterwoude	زاتروود ، زوترود ، زوتروود ، زویوترووود ، زیوترود
Zurink	زرینک ، زورینک ، زیورینک

Bibliography

N. AbdulJaleel and L. S. Larkey. Statistical transliteration for English-Arabic cross language information retrieval. In *Conference on Information and Knowledge Management*, pages 139–146, New Orleans, Louisiana, November 2003.

Y. Al-Onaizan and K. Knight. Machine transliteration of names in Arabic text. In *Proceedings of the ACL workshop on Computational approaches to semitic languages*, pages 1–13, Philadelphia, PA, July 2002a.

Y. Al-Onaizan and K. Knight. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 400–408, Philadelphia, PA, July 2002b.

Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. J. Och, D. Purdy, N. Smith, and D. Yarowsky. Statistical machine translation. Technical report, Johns Hopkins University, 1999.

I. Alegria, N. Ezeiza, and I. Fernandez. Named entities translation based on comparable corpora. In *Proceedings of the EACL Workshop on Multi-Word-Expressions in a Multilingual Context*, Trento, Italy, April 2006.

J. Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, San Francisco, 1995.

E. Aramaki, T. Imai, K. Miyo, and K. Ohe. Support vector machine based orthographic disambiguation. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation*, pages 21–30, Skovde, Sweden, September 2007.

E. Aramaki, T. Imai, K. Miyo, and K. Ohe. Orthographic disambiguation incorporating transliterated probability. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 48–55, Hyderabad, India, January 2008.

M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart. Algorithms for Arabic name transliteration. *IBM Journal of research and Development*, 38(2):183–194, 1994.

D. Arnold, L. Balkan, S. Meijer, L. Humphreys, and L. Sadler. *Machine Translation, An Introductory Guide*. NCC Blackwell Ltd., 2001.

S. Bangalore, G. Bordel, and G. Riccardi. Computing consensus translation from multiple machine translation systems. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 351–354, Kyoto,Japan, December 2001.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

S. Bilac and H. Tanaka. Improving back-transliteration by combining information sources. In *Proceedings of First International Joint Conference on Natural Language Processing*, volume 3248 of *Lecture Notes in Computer Science*, pages 216–223. Springer, 2004.

S. Bilac and H. Tanaka. Direct combination of spelling and pronunciation information for robust back-transliteration. In *Conferences on Computational Linguistics and Intelligent Text Processing*, pages 413–424, Mexico City, Mexico, February 2005.

P. F. Brown, J. Cocke, S. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computional Linguistics*, 19(2): 263–311, 1993.

P. F. Brown, J. Cocke, S. D. Pietra, V. J. D. Pietra, F. Jelinek, J. C. Lai, and R. L. Mercer. Method and system for natural language translation. *Expert Systems with Applications*, 11(2):IV–IV(1), 1996.

C. Butler. *Statistics in Linguistics*. Basil Blackwell, New York, 1985.

R. Catizone, G. Russell, and S. Warwick. Deriving translation data from bilingual texts. In *Proceedings of the First International Lexical Acquisition Workshop*, pages 1–7, Detroit, Michigan, August 1989.

C. Chen and H.-H. Chen. A high-accurate Chinese-English NE backward translation system combining both lexical information and web statistics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL on Main Conference Poster Sessions*, pages 81–88, Sydney, Australia, July 2006.

H.-H. Chen and J.-C. Lee. Identification and classification of proper nouns in Chinese texts. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 222–229, Copenhagen, Denmark, August 1996.

S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, June 1996.

P.-J. Cheng, Y.-C. Pan, W.-H. Lu, and L.-F. Chien. Creating multilingual translation lexicons with regional variations using web corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, page 534, Barcelona, Spain, July 2004.

D. Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan, June 2005.

J. G. Cleary and I. H. Witten. A comparison of enumerative and adaptive codes. *IEEE Transactions on Information Theory*, 30(2):306–315, 1984.

M. A. Covington. An algorithm to align words for historical comparison. *Computational Linguistics*, 22(4):481–496, 1996.

D. Crystal. *How Language Works: How Babies Babble, Words Change Meaning, and Languages Live or Die*. The Overlook Press, New York, 2006.

I. Dagan and K. Church. Termight: identifying and translating technical terminology. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, pages 34–40, Stuttgart, Germany, October 1994.

R. Dale. Language technology. In *Slides of HCSNet Summer School course*, Sydney, Australia, June 2007.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

M. Divay and A. J. Vitale. Algorithms for grapheme-phoneme translation for English and French: applications for database searches and speech synthesis. *Computational Linguistics*, 23(4):495–523, 1997.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons Inc., New York, 1973.

A. Ekbal, S. K. Naskar, and S. Bandyopadhyay. A modified joint source-channel model for transliteration. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL on Main Conference Poster Sessions*, Sydney, Australia, July 2006.

D. Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1998.

D. Freitag and S. Khadivi. A sequence alignment model based on the averaged perceptron. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 238–247, Prague, Czech Republic, June 2007.

P. Fung and K. McKeown. A technical word- and term-translation aid using noisy parallel corpora across language groups. *Machine Translation*, 12(1-2):53–87, 1997.

P. Fung and L. Y. Yee. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 414–420, Montreal, Canada, August 1998.

W. Gale and K. Church. Identifying word correspondance in parallel texts. In *Proceedings of the workshop on Speech and Natural Language*, pages 152–157, Pacific Grove, California, February 1991.

W. Gao, K.-F. Wong, and W. Lam. Phoneme-based transliteration of foreign names for OOV problem. In *proceedings of the 1st International Joint Conference on Natural Language Processing*, volume 3248 of *Lecture Notes in Computer Science*, pages 110–119. Springer, 2004a.

W. Gao, K.-F. Wong, and W. Lam. Improving transliteration with precise alignment of phoneme chunks and using contextual features. In *Information Retrieval Technology, Asia Information Retrieval Symposium*, volume 3411 of *Lecture Notes in Computer Science*, pages 106–117, Beijing, China, October 2004b.

U. Germann. Building a statistical machine translation system from scratch: how much bang for the buck can we expect? In *Proceedings of the workshop on Data-Driven Methods in Machine Translation*, pages 1–8, Toulouse, France, July 2001.

I. Goto, N. Kato, T. Ehara, and H. Tanaka. Back transliteration from Japanese to English using target English context. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 827–833, Geneva, Switzerland, July 2004.

D. Grossman and O. Frieder. *Information Retrieval; Algorithms and Heuristics*. Springer, 2004.

P. A. V. Hall and G. R. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, 1980.

H. Hassan, K. Simaán, and A. Way. Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 288–295, Prague, Czech Republic, June 2007.

J. C. Henderson and E. Brill. Exploiting diversity in natural language processing: combining parsers. In *Proceedings of the Fourth Conference on Empirical Methods in Natural Language Processing*, pages 187–194, College Park, Maryland, June 1999.

F. Huang. Cluster-specific named entity transliteration. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada, October 2005.

F. Huang and K. Papineni. Hierarchical system combination for machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 277–286, Prague, Czech Republic, June 2007.

F. Huang and S. Vogel. Improved named entity translation and bilingual named entity extraction. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, pages 253–258, 2002.

F. Huang, S. Vogel, and A. Waibel. Clustering and classifying person names by origin. In *Proceedings of National Conference on Artificial Intelligence*, pages 1056–1061, Pittsburgh, Pennsylvania, July 2005.

S. M. Humphrey, W. J. Rogers, H. Kilicoglu, D. Demner-Fushman, and T. C. Rindflesch. Word sense disambiguation by selecting the best semantic type based on journal descriptor indexing: Preliminary experiment. *Journal of the American Society for Information Science and Technology*, 57(1):96–113, 2006.

M. Hundt. *Corpus Linguistics and the Web (Language and Computers 59)*. Editions Rodopi BV, 2006.

N. Ide and J. Véronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24(1):1–40, 1998.

IPA. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press, 1999.

F. Jelinek. Fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675–685, 1969.

K. S. Jeong, S. H. Myaeng, J. S. Lee, and K. S. Choi. Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing and Management*, 35(4):523–540, 1999.

S. Y. Jung, S. L. Hong, and E. Paek. An English to Korean transliteration model of extended Markov window. In *Proceedings of the 18th Conference on Computational linguistics*, pages 383–389, Saarbrücken, Germany, August 2000.

D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2008.

J. Justeson and S. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27, 1995.

B.-J. Kang and K.-S. Choi. Automatic transliteration and back-transliteration by decision tree learning. In *Conference on Language Resources and Evaluation*, pages 1135–1411, Athens, Greece, May 2000.

I.-H. Kang and G. Kim. English-to-Korean transliteration using multiple unbounded overlapping phoneme chunks. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 418–424, Saarbrücken, Germany, August 2000.

M. Kashani, F. Popowich, and A. Sarkar. Automatic transliteration of proper nouns from Arabic to English. In *In Proceedings of the 2nd Workshop on Computational Approaches to Arabic Script-based Languages*, pages 81–87, Stanford,California, July 2007.

H. Keskustalo, A. Pirkola, K. Visala, E. Leppänen, and K. Järvelin. Non-adjacent digrams improve matching of cross-lingual spelling variants. In *String Processing and Information Retrieval*, volume 2857 of *Lecture Notes in Computer Science*, pages 252–265, Manaus, Brazil, October 2003.

A. Klementiev and D. Roth. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 817–824, Sydney, Australia, July 2006.

K. Knight. A statistical MT tutorial workbook, 1999. URL http://www.isi.edu/natural-language/mt/wkbk.rtf.

K. Knight and J. Graehl. Machine transliteration. In *Proceedings of the 8th Conference on European Chapter of the Association for Computational Linguistics*, pages 128–135, Madrid, Spain, July 1997.

K. Knight and J. Graehl. Machine transliteration. *Computational Linguistics*, 24(4):599–612, 1998.

P. Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of Conference of the Association for Machine Translation in the Americas*, pages 115–124, Washington DC, September 2004.

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Edmonton, Canada, May 2003.

W. Lam, R. Huang, and P.-S. Cheung. Learning phonetic similarity for matching named entity translations and mining new translations. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 289–296, Sheffield, UK, July 2004.

W. Lam, S.-K. Chan, and R. Huang. Named entity translation matching and learning: with application for mining unseen translations. *ACM Transactions on Information Systems*, 25(1):Article 2, 2007.

L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 289–297, Zurich, Switzerland, August 1996.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

H. Li, M. Zhang, and J. Su. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 159–166, Barcelona, Spain, July 2004.

H. Li, K. C. Sim, J.-S. Kuo, and M. Dong. Semantic transliteration of personal names. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 120–127, Prague, Czech Republic, June 2007.

W.-H. Lin and H.-H. Chen. Backward machine transliteration by learning phonetic similarity. In *Proceeding of the 6th Conference on Natural Language Learning*, pages 1–7, Taipei, Taiwan, August 2002.

K. Lindén. Multilingual modeling of cross-lingual spelling variants. *Information Retrieval*, 9 (3):295–310, 2005.

W.-H. Lu, L.-F. Chien, and H.-J. Lee. Translation of web queries using anchor text mining. *ACM Transactions on Asian Language Information Processing*, 1(2):159–172, 2002.

M. G. A. Malik. Punjabi machine transliteration. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 1137–1144, Sydney, Australia, July 2006.

T. Masuyama and H. Nakagawa. Web-based acquisition of japanese katakana variants. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 338–344, Salvador, Brazil, August 2005.

E. Matusov, N. Ueffing, and H. Ney. Computing consensus translation for multiple machine translation systems using enhanced hypothesis alignment. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–40, Trento, Italy, April 2006.

T. McEnery and A. Wilson. *Corpus Linguistics*. Edinburgh University Press, 1996.

I. D. Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.

Z. Min, L. Haizhou, and S. Jian. Direct orthographical mapping for machine transliteration. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 716, Geneva, Switzerland, August 2004.

T. Mitchell. *Machine Learning*. McGraw Hill, Columbus, 1997.

R. C. Moore. Improving IBM word-alignment Model 1. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 519–526, Barcelona, Spain, July 2004.

E. Y. Mun and A. V. Eye. *Analyzing Rater Agreement: Manifest Variable Methods*. Lawrence Erlbaum Associates, New Jersey, 2004.

M. Nagata, T. Saito, and K. Suzuki. Using the web as a bilingual dictionary. In *Proceedings of the workshop on Data-driven methods in machine translation*, pages 1–8, Toulouse, France, 2001.

T. Nomoto. Multi-engine machine translation with voted language model. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 494–501, Barcelona, Spain, July 2004.

S. Nowson and R. Dale. Charting democracy across parsers. In *Proceedings of the Australasian Language Technology Workshop*, pages 75–82, Melbourne, Australia, December 2007.

F. J. Och and H. Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong, China, October 2000.

F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

J.-H. Oh and K.-S. Choi. An English-Korean transliteration model using pronunciation and contextual rules. In *Proceedings of the 19th International Conference on Computational linguistics*, Taipei, Taiwan, August 2002.

J.-H. Oh and K.-S. Choi. Machine learning based English-to-Korean transliteration using grapheme and phoneme information. *IEICE Transactions on Information and Systems*, E88-D(7):1737–1748, 2005.

J.-H. Oh and K.-S. Choi. An ensemble of transliteration models for information retrieval. *Information Processing and Management*, 42(4):980–1002, 2006.

J.-H. Oh and H. Isahara. Machine transliteration using multiple transliteration engines and hypothesis re-ranking. In *Proceedings of the 11th Machine Translation Summit*, pages 353–360, Copenhagen, Denmark, September 2007a.

J.-H. Oh and H. Isahara. Validating transliteration hypotheses using the web: web counts vs. web mining. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 267–270, Silicon Valley, CA, November 2007b.

J.-H. Oh, K.-S. Choi, and H. Isahara. A machine transliteration model based on correspondence between graphemes and phonemes. *ACM Transactions on Asian Language Information Processing (TALIP)*, 5(3):185–208, 2006a.

J.-H. Oh, K.-S. Choi, and H. Isahara. Improving machine transliteration performance by using multiple transliteration models. In *Proceedings of the 21st International Conference on Computer Processing of Oriental Languages*, pages 85–96, Singapore, December 2006b.

W. Pang, Z. Yang, Z. Chen, W. Wei, B. Xu, and C. Zong. The CASIA phrase-based machine translation system. In *Proceedings of the International Workshop on Spoken Language Translation*, 8pp, Pittsburgh, PA, October 2005.

J. Pearson. *Terms in Context*. John Benjamins Publishing Company, 1998.

T. Pedersen. A simple approach to building ensembles of Naive Bayesian classifiers for word sense disambiguation. In *Proceedings of the 1st Conference on North American Chapter of the Association for Computational Linguistics*, pages 63–69, Seattle, Washington, May 2000.

A. Pirkola, J. Toivonen, H. Keskustalo, and K. Järvelin. FITE-TRT: a high quality translation technique for OOV words. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 1043–1049, Dijon, France, April 2006.

R. Rapp. Identifying word translations in non-parallel texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 320–322, Cambridge, Massachusetts, June 1995.

A.-V. Rosti, N. F. Ayan, B. Xiang, S. Matsoukas, R. Schwartz, and B. Dorr. Combining outputs from multiple machine translation systems. In *The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, April 2007.

D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1136–1142, Montreal, Canada, August 1998.

G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

T. Sherif and G. Kondrak. Substring-based transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 944–951, Prague, Czech Republic, June 2007a.

T. Sherif and G. Kondrak. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 864–871, Czeck Republic, June 2007b.

R. Sproat, T. Tao, and C. X. Zhai. Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 73–80, Sydney, Australia, July 2006.

B. Stalls and K. Knight. Translating names and technical terms in Arabic text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*, pages 34–41, Montreal, Canada, August 1998.

T. Talvensaari, K. Järvelin, and M. Juhola. Creating and exploiting a comparable corpus in cross-language information retrieval. *ACM Transactions on Information Systems*, 25(1), 2007.

J. Toivonen, A. Pirkola, H. Keskustalo, K. Visala, and K. Järvelin. Translating cross-lingual spelling variants using transformation rules. *Information Processing and Management*, 41 (4):859–872, 2005.

K. Toutanova, H. T. Ilhan, and C. D. Manning. Extensions to HMM-based statistical word alignment models. In *Proceedings of the Conference on Empirical methods in Natural Language Processing*, pages 87–94, Pennsylvania, Philadelphia, July 2002.

P. Van der Eijk. Automating the acquisition of bilingual terminology. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113–119, Utrecht, The Netherlands, April 1993.

H. van Halteren, J. Zavrel, and W. Daelemans. Improving data driven word class tagging by system combination. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 491–497, Montreal, Canada, August 1998.

D. Vickrey, L. Biewald, M. Teyssier, and D. Koller. Word-sense disambiguation for machine translation. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 771–778, Vancouver, Canada, October 2005.

P. Virga and S. Khudanpur. Transliteration of proper names in cross-language applications. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 365–366, Toronto, Canada, July 2003a.

P. Virga and S. Khudanpur. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL Workshop on Multilingual and Mixed-Language Named Entity Recognition*, pages 57–64, Sapporo, Japan, July 2003b.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

S. Vogel, H. Ney, and C. Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational linguistics*, pages 836–841, Copenhagen, Denmark, August 1996.

S. Wan and C. Verspoor. Automatic English-Chinese name transliteration for development of multilingual resources. In *Proceedings of the 17th International Conference on Computational linguistics*, pages 1352–1356, Montreal, Canada, July 1998.

L. Xu, A. Fujii, and T. Ishikawa. Modeling impression in probabilistic transliteration into chinese. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 242–249, Sydney, Australia, July 2006.

D. Zelenko and C. Aone. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 612–617, Sydney, Australia, July 2006.

Y. Zhang, F. Huang, and S. Vogel. Mining translations of OOV terms from the web through cross-lingual query expansion. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 669–670, Salvador, Brazil, August 2005.